ＭＹ－ＩＣＥ ＡＥ
Ｈ８／３００Ｈシリーズ
モニタ ｆｏｒ Ｗｉｎｄｏｗｓ ９５／ＮＴ
オペレーションマニュアル

株式会社　日立超ＬＳＩシステムズ
ＭＰ３０５２ＡＥ５－ＷＰＭ

はじめに

　このたびは、ＭＹ－ＩＣＥモニタ　ｆｏｒ　Ｗｉｎｄｏｗｓ（以下、モニタと略します）をお買い上げいただき誠にありがとうございます。この取扱説明書は、ＭＹ－ＩＣＥモニタの機能について説明してあります。

　モニタは、ＭＳ－Ｗｉｎｄｏｗｓ９５／ＮＴのもとで動作します。

　技術的なサポートは次の直通電話等が用意されていますのでご利用ください。

直通電話番号　：　0120-25-3047
ＦＡＸ　　　　：　042-351-6617

・この取り扱い説明書はＨ８／３００Ｈシリーズのモニタに関して説明してあります。
・ＰＣ－９８０１、ＰＣ－９８２１はＮＥＣ社の商標です。
・ＰＣ／ＡＴはＩＢＭ社の商標です。
・ＭＳ－ＤＯＳ、ＭＳ－Ｗｉｎｄｏｗｓはマイクロソフト社の商標です。
・提供されますＣＤ－ＲにＲＥＡＤＭＥ．ＤＯＣというファイルがある場合は追加の補足説明が記述されていますのでお読みください。
・本製品の内容は機能向上や改良のため、予告無しに変更されるときがありますのでご了承ください。
・本製品は、万全の注意を払って作成されていますが、ご利用になった結果については、（株）日立超ＬＳＩシステムズは一切の責任を負いかねますのでご了承ください。
・本製品に関するご意見、ご感想、ご要望等がありましたら、弊社までおよせください。将来のバージョンアップ等の参考にさせていただきます。

目　次

１．特長

主な特長を次に挙げます。

(1) ＭＤＩインタフェース

モニタは、ＭＤＩ（Ｍｕｌｔｉｐｌｅ Ｄｏｃｕｍｅｎｔ Ｉｎｔｅｒｆａｃｅ）を採用しています。これにより複数のドキュメントウィンドウ（ソース、コマンド、レジスタ等のウィンドウ）を統一して扱うことができます。下図にウィンドウの表示例を挙げます。

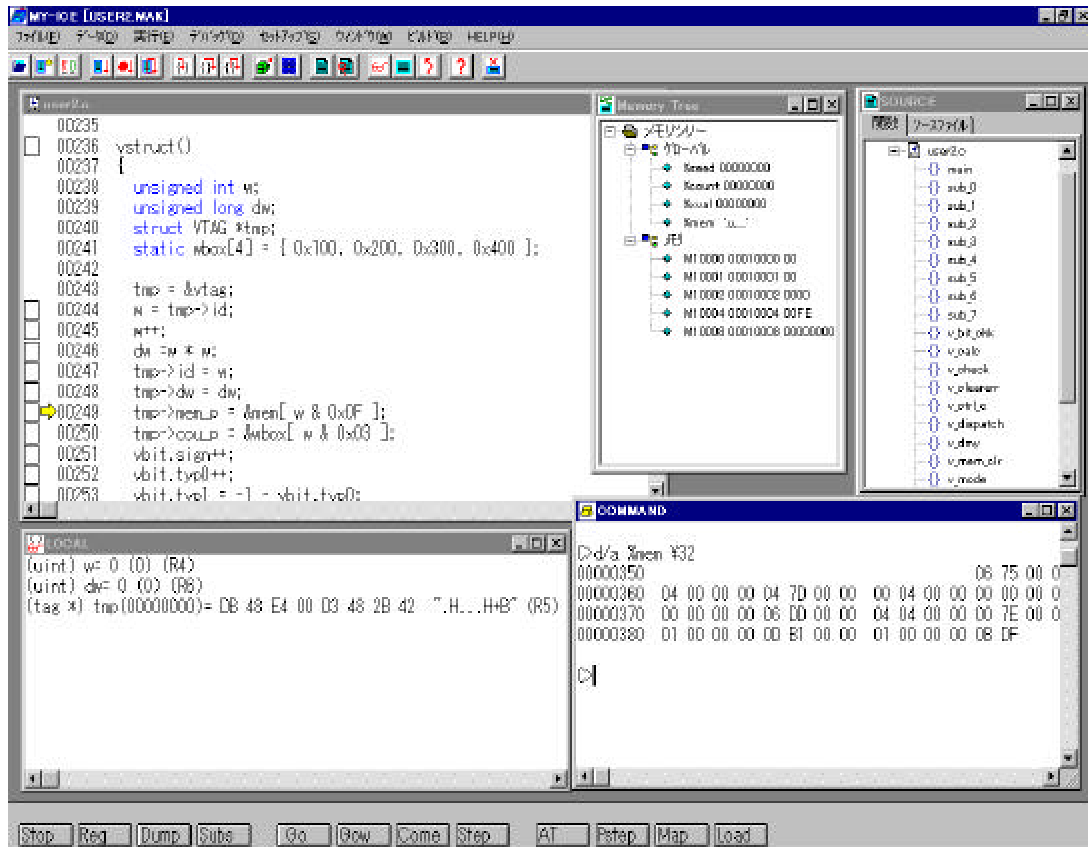


図　１－１　　モニタの表示例

(2) ソースコードデバッキング

ソースウィンドウには、お客様がコーディングされたＣ言語、またはアセンブラのソースプログラムをそのまま表示します。直接、ソースプログラム上での一行単位でのステップ実行や、ブレークポイントの設定が可能です。

(3) インスペクト機能

構造体の変数を表示可能なインスペクトウィンドウをサポートしています。また、ソースウィンドウの変数でマウスポインタを静止しますと、その変数の値を表示するチップインスペクト機能があります。

(4) ビルド機能

ユーザプログラムを構成するソースファイル群の中で変更のあったソースファイルを自動的にコンパイル、およびアセンブルしてユーザプログラムを新たに作成するビルド機能があります。

(5) Ｉ／Ｏレジスタ表示機能

Ｉ／Ｏレジスタの内容をツリービュー形式で表示します。同様の形式でメモリ内容も表示可能です。

(6) カスタマイズ機能

ソースウィンドウ、混在モードウィンドウのテキスト表示において表示色のカスタマイズが可能です。また、ツールバーのカスタマイズも可能です。

(7) ソースファイル、関数一覧表示機能

ユーザプログラムを構成するソースファイルと関数の一覧をツリービュー形式で表示します。

(8) ダイナミックコンフィグレーション

モニタ起動時、または、モニタ起動後に、使用するＩＣＥやターゲットＣＰＵを指定して、ＩＣＥのコンフィグレーションをすることができます。

２．動作環境とインストール

2.1 動作環境

モニタを動作させるには次の環境が必要です。

パソコン：　ＰＣ／ＡＴおよび互換機、ＰＣ９８２１シリーズ
ＣＰＵ　：　８０４８６以上
メモリ　：　３２MB以上
ＯＳ　　：　ＰＣ／ＡＴはＷｉｎｄｏｗｓ９５／９８、ＷｉｎｄｏｗｓＮＴ（Ｖｅｒ ４．０）
　　　　　　ＰＣ９８２１はＷｉｎｄｏｗｓ９５／９８

2.2 インストール手順

提供されます一枚目のCD-Rの SETUP.EXE を実行してください。
インストール作業は、インストールプログラムの指示に従って実行してください。

2.3 制限事項

(1) 日立製コンパイラ以外のサポートに関して

日立製以外のコンパイラが出力するロードモジュールフォーマットを、モニタではロードすることはできません。

(2) 混在モードウィンドウの縦のスクロールバーの使用に関して

ユーザプログラム実行中（GOまたはGORESコマンド実行中）の混在モードウィンドウの縦のスクロールバーは動作しません。

(3) 使用中のＭＹ－ＩＣＥに対するモニタの再起動に関して

ご使用中（モニタ起動中）のＭＹ－ＩＣＥに対して、モニタを再起動しないでください。

(4) ビルド実行中に関して

ビルド実行中は「ファイル」、「実行」、「セットアップ」に関するメニューは使用不可能になります。

(5) エディタの使用に関して

エディタでソースファイルを修正中の時は、モニタのビルド、リビルドコマンドはを実行する前に、ソースファイルをセーブしてください。また、現在のバージョンのモニタでは、エディタの機能をサポートしていません。

(6) ロングファイル名を使用したソースファイルの表示に関して

Windows95/NTのロングファイル名を使用したソースファイルは表示できません。
この場合、ショートファイル名で再ビルドしますと表示可能となります。
ショートファイル名はOSのMS-DOSプロンプトまたはコマンドプロンプトのDIRコマンドで表示できます。(WindowsNTでは/Xオプションが必要です)

(7) オペレーションサイズに関する制限事項

以下のコマンドにおいてワード、ロングワードによるオペレーションは指定できません。
指定サイズによるオペレーションが必要となる場合はSUBS、FILL、DUMP、CHNGコマンドをご使用下さい。

指定できないコマンド SRCH,LOAD,SAVE,MOVE,DASM

2.4 コンフィグレーションの設定

2.4.1 ダイナミックコンフィグレーション

ダイナミックコンフィグレーション機能について説明します。

(1) 概要

モニタには、ダイナミックコンフィグレーション機能を追加しています。この機能は、モニタ起動時、または、起動後にＭＹ－ＩＣＥの設定を動的に変更するためのものです。設定の変更は、［コンフィグレーション］のダイアログボックスで行います。

このダイアログボックスで指定できる項目と内容を、表2-１に示します。

表2-１　［コンフィグレーション］ダイアログボックスの項目と内容

項目	内容
接続	モニタで接続するＩＣＥをリストボックスの中から選択してください。リストボックスには、モニタで使用可能なＩＣＥのみが表示されます。
ＣＰＵ	ターゲットＣＰＵを選択します。指定可能なＣＰＵ名がリストに表示されます。
Ｉ／Ｏレジスタファイル	Ｉ／Ｏレジスタウィンドウを表示するためのデータファイルを選択してください。指定可能なファイル名が「リストに表示されます。
ＳＹＭＢ	デバッグ情報を格納するためのバッファサイズを、ＫＢ単位で指定してください。

(2) モニタ起動時のコンフィグレーション

モニタをはじめて起動する場合や、前回の起動に失敗している場合、および、パソコンにインストールされているＩ／ＦカードやＩＣＥに変更がある場合は、起動時に［コンフィグレーション］ダイアログが表示されます。起動時に、このダイアログが表示された場合は、すべての項目を指定して［ＯＫ］ボタンを選択してください。

１つでも指定されていない項目がある場合は［ＯＫ］ボタンを選択してもコンフィグレーションできません。

また、［キャンセル］ボタンを選択した場合は、モニタの起動を終了します。

コンフィグレーションに成功した場合、コンフィグレーションデータは、プロファイルに保存されます。以降のモニタの起動時には、このコンフィグレーションデータが使用されます。

(3) モニタ起動後のコンフィグレーション

起動後にコンフィグレーションを行う場合、または、現在のコンフィグレーションを変更する場合には、［セットアップ］－［コンフィグレーション］コマンドを選択してください。［コンフィグレーション］ダイアログが表示されますので、変更したい項目を指定して［ＯＫ］ボタンを選択してください。

［キャンセル］ボタンを選択すると、コンフィグレーションは行われません。

(4) 起動時のエラーについて

起動時に、「このモニタソフトで接続可能なICEは見つかりませんでした」のエラーメッセージが表示された場合には、以下の項目について調べてください。

・ＩＣＥ本体の電源が入っているか。
・ＩＣＥとＩ／Ｆカードが正しく接続されているか。
・モニタで使用可能なＩＣＥがＩ／Ｆカードに接続されているか。
・ドライバが正しくインストールされ動作しているか。

2.4.2 コンフィグレーションファイル

コンフィグレーションファイル ＩＣＥＣＯＮＦ．ＤＡＴ によりハードウエア条件を指定できます。指定できる項目を以下に説明します。

(1) ＢＲＥＱ／
プログラム停止中のＢＲＥＱ入力の有効、無効を次のように指定します。

Ｅ：有効、Ｄ：無効

（注）プログラム停止中のＢＲＥＱ入力を有効にした場合、LINEコマンドのBREQ設定にはかかわらず
プログラム実行中のＢＲＥＱ入力が常に有効になります。

(2)ＰＥＲＣＬＫ／ では、パフォーマンス測定のクロック源を指定します。
・バスクロックを使用する場合：周波数を指定（ＭＨｚ）
・内部１６ＭＨｚを使用する場合：S16を指定
・内部８ＭＨｚを使用する場合：S8を指定
・内部１ＭＨｚを使用する場合：S1を指定

上記以外の項目は、変更しないでください。

2.5 提供されるプログラム等

本モニタが提供するプログラム、およびデータを次にあげます。

ＩＣＥ．ＥＸＥ	ＩＣＥモニタのプログラム本体です。
ＩＣＥＳＹＭ．ＥＸＥ	デバッグ情報ファイルを作成します。 「ＩＣＥＳＹＭ　コマンド」を参照してください。
ＩＣＥＣＯＮＦ．ＤＡＴ	ＩＣＥとＰＣとの接続情報を定義しているコンフィグレーションファイルです。
ＩＣＥＩＮＩＴ．ＤＡＴ	ＩＣＥモニタ立ち上げ時の、初期化用コマンドファイルです。 本ファイルにより、コマンドの短縮形、およびファンクションキーの設定の初期値が定義されています。本ファイルの内容の一部を変更することによりお客様の好みに合った設定にすることが可能です。変更するときは、コメントの内容に従って変更可能なところのみ変更するようにしてください。
ＦＩＲＭＨ８．ＤＡＴ ＩＣＥＦＬＥＸ１．ＤＡＴ ＩＣＥＦＬＥＸ２．ＤＡＴ	エミュレーション制御用プログラムです。
ＥＲＲＩＣＥ．Ｈ	エラーメッセージファイルです。
ＩＣＥＣＭＤ．ＤＡＴ ＩＣＥＡＬＩＡＳ．ＤＡＴ	システム用コマンドファイルです。
ＩＯＲＥＧ．ＤＡＴ （ＩＯｘｘｘｘ．ＤＡＴ）	Ｉ／Ｏレジスタウィンドウ用内蔵レジスタの定義ファイルです。 （対象ＣＰＵで内蔵レジスタが異なるときに他の定義ファイルが提供されます。ｘｘｘｘは品種名を示します。）
ＭＥＭＴＲＥＥ．ＤＡＴ	メモリツリーウィンドウの定義ファイルです。
ＵＳＥＲ２．Ｃ ＵＳＥＲ２．ＡＢＳ ＵＳＥＲ２．ＩＣＥ	サンプル用ユーザプログラムファイルです。

Ｉ／Ｏレジスタウィンドウ、メモリツリ－ウィンドウに関しましては、「4.4 ドキュメントウィンドウ」を参照してください。

３．ユーザプログラムの作成手順

モニタは、次のコンパイラ、およびアセンブラをサポートしています。

- 日立製Ｃコンパイラ、アセンブラ
  Ｈ８Ｓ、Ｈ８／３００シリーズＣコンパイラ
  Ｈ８Ｓ、Ｈ８／３００シリーズクロスアセンブラ
  Ｈシリーズリンケージエディタ
  （リンカはアセンブラパッケージに付属します。）

ユーザプログラムの作成にはモニタのビルド機能を用いる方法と、ＤＯＳ窓等でコンパイラ、アセンブラ、リンケージエディタを実行する方法があります。

「３．１　ビルド機能によるユーザプログラム作成手順」ではビルド機能を用いたユーザプログラムの作成手順を示します。ビルド機能の詳細に関しましては「４．２．１ ビルド機能」を参照してください。
ビルド機能を使用するには、Ｗｉｎｄｏｗｓ版のコンパイラ、アセンブラ、リンケージエディタが必要です。

「３．２　ＤＯＳ窓でのユーザプログラム作成手順」ではＤＯＳ窓でのユーザプログラムの作成手順を示します。ＤＯＳ版のコンパイラ、アセンブラ、リンケージエデイタを使用した例です。

3.1　ビルド機能によるユーザプログラム作成手順

提供されますサンプルユーザプログラムUSER2.Cをビルド機能でコンパイルする手順を示します。

［ビルド］－［新規作成］を選択します。
［新規プロジェクトファイルを開く］のダイアログボックスがオープンします。ダイアログボックス内の［ファイルの場所］をUSER2.Cのファイルが存在するディレクトリに移動します。プロジェクトファイル名の入力が必要ですのでこの例ではTEST.MAK をキー入力して［開く］を選択します。［プロジェクトへ追加する］のダイアログボックスがオープンしますので、USER2.Cを選択して［ＯＫ］を選択します。
以上の作業でプロジェクトファイルTEST.MAKにUSER2.Cのユーザプログラムが追加されます。
（［ビルド］－［編集］で確認することができます。）

［ビルド］－［オプション設定］でオプションの設定のダイアログボックスがオープンします。
［Ｃコンパイラ］のタグコントロールの［ＣＰＵ種別］でＳＨ２を選択します。
［リンカ］、［環境設定］のタグコントロールと各項目の設定例を下記に示します。

<タグ名>	<項目名>	<設定例>	<意味>
リンカ	ライブラリファイル	D:\CH38\LIB\C38HA.LIB	ライブラリファイルを指定します。
環境設定	コンパイラ	D:\CH38\BIN	コンパイラの実行ファイルのパス名を指定します。
環境設定	アセンブラ	D:\H8S_Assembler\Assembler	アセンブラの実行ファイルのパス名を指定します。
環境設定	リンカ	D:\H8S_Assembler\LinkageEditor	ＬＮＫの実行ファイルのパス名を指定します。
環境設定	ＩＣＥＳＹＭ	D:\WICE	ＩＣＥＳＹＭの実行ファイルのパス名を指定します。

［ビルド］－［ビルド］で、コンパイル、リンク、ＩＣＥＳＹＭ、ロードまで自動的に実行します。
アセンブラもアセンブラソースファイルを選択することにより同様に実行されます。

3.2　ＤＯＳ窓でのユーザプログラムの作成手順

3.2.1 日立製Ｃコンパイラのコンパイル手順

日立製Ｃコンパイラを使用したときのコンパイル等の手順を下記に示します。
<path>　はディレクトリ名を、<file>　はファイル名を表します。

(1) コンパイル

C>ch38 /debug /cpu=300ha <file>
　　debug オプションを指定することにより、デバック時に使用する各種デバック
　　情報を出力します。

(2) リンク

C>lnk <file> /debug /lib=a:\ch38\lib\c38ha.lib [/entry=<シンボル名>]
　　entry オプションで実行開始アドレスを指定します。シンボル名を使用する
　　ときは、先頭にアンダーラインを付加してください。
　　例)　/entry=_main
　　ＩＣＥモニタはロード（ＬＯＡＤコマンド）直後、プログラムカウンタを
　　entry のアドレスに設定します。

C>icesym <file>
　　icesymコマンドによりデバック情報ファイル（拡張.ice）を作成します。
　　詳細は「ＩＣＥＳＹＭコマンド」を参照してください。

(3) ロード
ＩＣＥモニタを起動します。
>load <file>
　　ロード終了後、画面は entry 位置のソースプログラムを表示します。

(4) 使用例
ソースファイルを sample.c とした場合の使用例を示します。

C>ch38 /debug /cpu=300ha sample.c
　　生成ファイル　　sample.obj　sample.lst
C>lnk sample.obj /debug /entry=_main /lib=a:\ch38\lib\c38ha.lib
　　ライブラリを使用するときは、 /lib のオプションでライブラリを指定
　　します。
　　生成ファイル　　sample.abs
C>icesym sample
　　生成ファイル　　sample.ice

ICEモニタを起動します。
>icesrc a:\src　ソースファイルのあるパスを指定します。
>set mode/1
>map 0 ffff eram
　...................
>load sample
　　ロード終了後の画面には、関数 main() の位置のソースプログラムを
　　表示します。

3.2.2 日立製アセンブラのアセンブル手順

日立製アセンブラを使用したときのアセンブル等の手順を下記に示します。

(1) アセンブル

C>asm38 <file> /debug /cpu=300ha
debug オプションを指定することにより、デバック時に使用する各種デバック情報を出力します。

(2) リンク
C>lnk <file> /debug [/entry=<シンボル名>]
アセンブラプログラム中の任意の外部定義シンボル名を /entry に設定できます。
ＩＣＥモニタのＬＯＡＤコマンドは、プログラムカウンタを/entryのアドレスに設定します。

C>icesym <file>
icesymコマンドによりデバック情報ファイル（拡張.ice）を作成します。
詳細は「ＩＣＥＳＹMコマンド」を参照してください。

(3) ロード
ＩＣＥモニタを起動します。

>icesrc <path>　ソースファイルのあるパスを定義してください。
>load <file>
load は、ＩＣＥモニタのコマンドです。
ロード終了後の画面には、/entry 位置のソースプログラムを表示します。

(4) 使用例
ソースファイルを sample.src とした場合の使用例を示します。

C>asm38 sample.src /debug /cpu=300ha
生成ファイル　sample.obj、sample.lst

C>lnk sample.obj /debug /entry=start
生成ファイル　sample.abs

C>icesym sample
生成ファイル　sample.ice

ＩＣＥモニタを起動します。
>icesrc a:\src　ソースファイルのあるパスを指定します。

>set mode/1
>map 0 ffff eram
....
>load sample
ロード終了後の画面には、シンボル名 start の位置のソースプログラムを表示します。

4．表示形式

4.1 ウィンドウ形式

モニタでは、図　4－1　の様なウィンドウ形式となります。

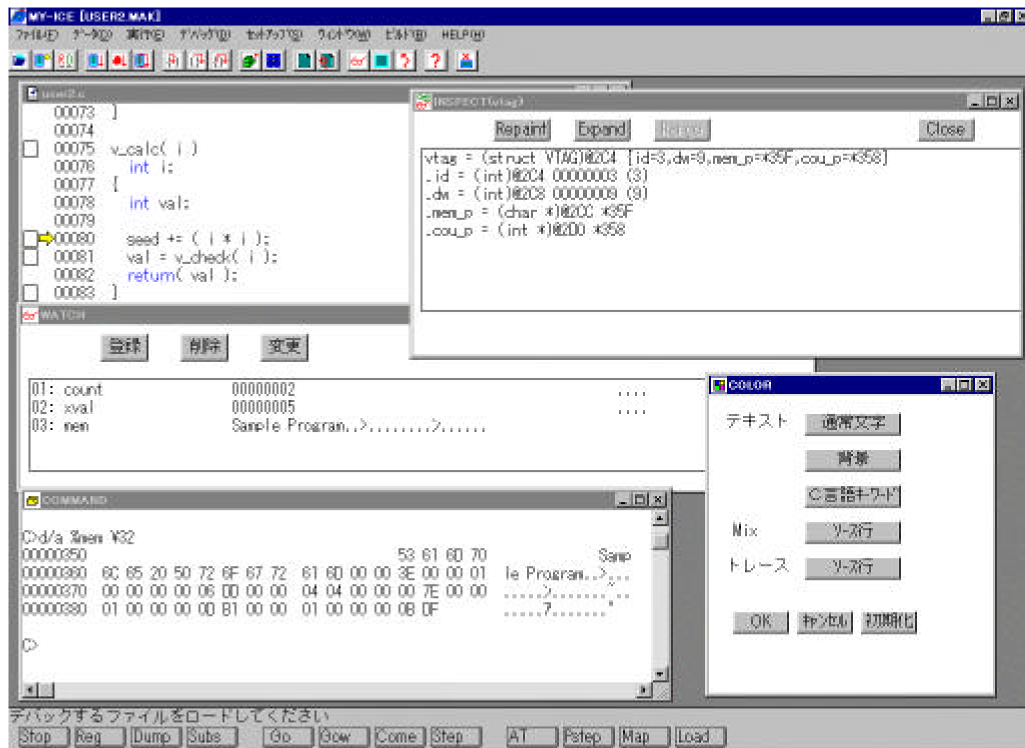


図　4－1　ウィンドウ形式

各ウィンドウの名称を図４－２に示します。

メニューバー
ツールバー
ドキュメントウィンドウ
ステータスバー
ファンクションキーバー

メニューバー	：	モニタが持つメニュー名が表示されます（4.2 メニューバー」参照）。
ツールバー	：	良く使用されるコマンドを登録してあります（「4.3 ツールバー」参照）。
ドキュメントウィンドウ	：	ドキュメントウィンドウとしては次のものがあります。 ソース、混在 モード、コマンド、テキスト、ダンプ、ウオッチ、ローカル、バックトレース、CPUステータス、レジスタ、スタック、トレースダンプ、シンボル数式計算、マップ（「4.4 ドキュメントウィンドウ」参照）
ステータスバー	：	ステータスを表示します。
ファンクションキーバー	：	ファンクションキーの表示

図　4－2　ウィンドウの名称

4.2 メニューバー

表４－１にメニュー名と、対応するドロップダウンメニュー（コマンドのリスト）を表します。

表４－１　メニュー

メニュー名	コマンドのリスト	機能
ファイル	ロード セーブ ベリファイ ソース 混在 コマンド オープン (ロードモジュール名) (ﾌﾟﾛｼﾞｪｸﾄﾌｧｲﾙ名) 終了	ユーザプログラムのロード ユーザプログラムのセーブ ユーザプログラムのベリファイ ソースウィンドウの表示 混在モードウィンドウの表示 コマンドウィンドウの表示 テキストウィンドウの表示 最近ロードしたモジュール名 最近ロードしたプロジェクトファイル名 モニタの終了
データ	インスペクト ダンプ ウオッチ ローカル バックトレース ＣＰＵステータス レジスタ スタック Ｉ／Ｏレジスタ メモリツリー	インスペクトウィンドウの表示 ダンプウィンドウの表示 ウオッチウィンドウの表示 ローカルウィンドウの表示 バックトレースウィンドウの表示 ＣＰＵステータスウィンドウの表示 レジスタウィンドウの表示 スタックウィンドウの表示 Ｉ／Ｏレジスタウィンドウの表示 メモリツリーウィンドウの表示
実行	実行 ステップ 関数ステップ リターン リセット ストップ	ユーザプログラムの実行 ステップ実行 関数ステップ実行 関数の呼び出しもとへの復帰 レジスタの初期化 ユーザプログラム実行の中止
デバッグ	シンボル 数式計算 トレース設定 トレースダンプ トレース削除 ソフトブレーク設定 ＰＣレーク設定 バスブレーク設定 バスブレーク削除 タイマ設定 タイマ削除 カバレージ設定 カバレージ表示 パフォーマンス設定 パフォーマンス表示	シンボルウィンドウの表示 数式計算ウィンドウの表示 トレース条件の設定 トレースダンプウィンドウの表示 トレース条件の削除 設定中のソフトブレーク条件の表示 設定中のＰＣブレーク条件の表示 バスブレーク条件の設定 バスブレーク条件の削除 タイマ条件の設定 タイマ条件の削除 カバレージ測定条件の設定 カバレージウィンドウの表示 パフォーマンス測定条件の設定 パフォーマンスウィンドウの表示
セットアップ	コンフィグレーション カスタマイズ マップ モード 信号制御 ﾂｰﾙﾊﾞｰｵﾝ/ｵﾌ ｽﾃｰﾀｽﾊﾞｰｵﾝ/ｵﾌ	ＩＣＥのコンフィグレーション 表示色、ツールバーのカスタマイズ マップウィンドウの表示 動作モードの設定 外部信号の制御 ツールバーの表示、非表示の選択 ステータスバーの表示、非表示の選択
ビルド	新規作成 開く 編集 オプション設定 閉じる コンパイル アセンブル ビルド リビルド 依存関係の調査 全ての依存関係の調査	プロジェクトの新規作成 プロジェクトのオープン プロジェクトの編集 コンパイラ等のオプション設定 プロジェクトを閉じる コンパイルの実行 アセンブルの実行 ビルドの実行（変更ソースをコンパイル） ビルドの実行（全ソースをコンパイル） ソースファイルの依存関係の調査、更新 全ソースファイルの依存関係の調査、更新
ヘルプ	バージョン情報	モニタのバージョン表示

各コマンドの実行において留意点を下記にあげます。その他に関しましては「4.4 ドキュメントウィンドウ」を参照してください。

・ロード、セーブ、ベリファイコマンド

これらのコマンドを選択しますと、ファイル選択のダイアログボックスがオープンします。

ロード、ベリファイコマンドでは、拡張子名とロードモジュールのフォーマットの対応は下記の通りです。

拡張子名		ロードモジュールフォーマット
ＡＢＳ	・・・	ＳＹＳＲＯＦフォーマット
ＳＡ	・・・	ＳＴＹＰＥフォーマット
ＭＯＴ	・・・	ＳＴＹＰＥフォーマット
ＢＩＮ	・・・	バイナリフォーマット

セーブコマンドでは、作成するファイル名、作成するロードモジュールのフォーマット（ラジオボタン）、セーブするアドレスを指定します。
（「LOAD コマンド」、「SAVE コマンド」、「VRFY コマンド」参照）

・ロードモジュール名

最近ロードしたロードモジュール名が最大４個まで表示されます。本ロードモジュール名を選択しますと、前回の動作環境を設定後、前回の画面を回復します。
「コンフィグレーションの設定」で指定されたプロファイルが使用されます。回復されるウィンドウは、ソース、混在モード、コマンド、テキスト、ダンプ、ウオッチ、ローカル、ＣＰＵステータス、レジスタ、スタック、プロジェクト、Ｉ／Ｏレジスタ、メモリツリーの各ウィンドウです。

ロードモジュール名はプロジェクトファイルがクローズされている状態のときに、ロードが実行される毎に順次登録されます。

・プロジェクトファイル名

最近ロードしたプロジェクトファイル名（拡張子名ＭＡＫ）が最大４個まで表示されます。本プロジェクトファイル名を選択しますと、前回の動作環境（ビルドに関する動作環境を含む）を設定後、前回の画面を回復します。
「コンフィグレーションの設定」で指定されたプロファイルが使用されます。回復されるウィンドウは、ソース、混在モード、コマンド、テキスト、ダンプ、ウオッチ、ローカル、ＣＰＵステータス、レジスタ、スタック、プロジェクト、Ｉ／Ｏレジスタ、メモリツリーの各ウィンドウです。

プロジェクトファイル名は、プロジェクトファイルがロードされる毎に順次登録されます。

・実行コマンド

ＧＯコマンド、ＧＯＷコマンド、ＧＯＲＥＳのいずれかを選択します。
（「GO コマンド」、「GOW コマンド」、「GORES コマンド」参照）

・トレース設定

トレース条件の設定として、ＡＬＬ（プログラム実行と同時にトレース開始）、ＳＭＰ（指定したトリガ条件でトレース開始）、ＲＮＧ（開始、中断、停止のトリガ条件を指定）が選択できます。（「TRC コマンド」参照）

・トレース表示

トレースダンプの表示条件（トレース開始フレーム番号、表示するトレースの行数、ある文字列を含む行のみの表示）設定と、トレースダンプウィンドウの表示が行えます。

・バスブレーク設定

バスブレーク０から３までが選択できます。（「BB コマンド」参照）

・タイマ設定

プログラム実行と同時にタイマを起動するか、起動、停止のトリガ条件を指定するのかが選択できます。（「TIM コマンド」参照）また、現在のタイマの値も表示されます。

・カバレージ設定

本メニューを選択しますとカバレージ設定のダイアログボックスがオープンします。

追加を選択しますとカバレージの測定範囲を登録できます。登録済みの項目を選択して変更を選択しますと測定範囲を変更できます。登録済みの項目を選択して削除を選択しますと、指定した項目を削除します。計測結果クリヤを選択しますと現在の測定結果をクリヤします。終了を選択しますとカバレージの測定条件を登録します（「COV コマンド」参照）。

・パフォーマンス設定
　本メニューを選択後、モジュール範囲の設定か、実行時間の分布測定の設定かを選択します。
　モジュール範囲を選択しますと実行時間、実行回数測定設定用のダイアログボックスがオープンします。設定項目を選択後、設定を選択しますと測定範囲を設定できます。設定項目を選択して削除を選択しますと、指定した項目を削除します。終了を選択しますとパフォーマンスの測定条件を登録します。
　実行時間分布測定を選択しますと、測定設定用のダイアログボックスがオープンします。測定するモジュールの先頭番地、最終番地、および、測定時間の最小時間、最大時間を指定します（「PER コマンド参照」）。

・カスタマイズ
　カスタマイズ機能としてはソースウィンドウ等の表示する色の設定とツールバーの設定が行えます。

　［色］のメニューを選択しますと表示色を設定するウィンドウが開きます。次に、各ボタンと変更されるウィンドウの表示色の対応を示します。

［テキスト］の［通常文字］・・・・・・ソースウィンドウとテキストウィンドウの表示文字の色
［テキスト］の［背景］・・・・・・・・ソースウィンドウとテキストウィンドウの背景色
［テキスト］の［Ｃ言語キーワード］・・ソースウィンドウとテキストウィンドウのＣ言語のキーワード
［Ｍｉｘ］　の［ソース行］・・・・・・混在モードウィンドウのソース行の色
［トレース］の［ソース行］・・・・・・トレースダンプウィンドウのソース行の色

［ツールバー］を選択しますと［ツールバーの変更］のダイアログボックスが開きます。［利用できるボタン］、［ツールバーのボタン］内のボタンをドラッグアンドドロップすることによりツールバーのカスタマイズが行えます。
　また、次の方法でも変更可能です。

・ボタンの追加
　［利用できるボタン］から、追加したいボタンを選択します。追加したい位置を、［ツールバーのボタン］で選択します。［追加］をクリックします。
・ボタンの削除
　［ツールバーのボタン］で、削除したいボタンを選択します。［削除］をクリックします。
・ボタンの移動
　［ツールバーのボタン］で、移動したいボタンを選択します。ボタンを移動したい方向に［上へ］［下へ］をクリックします。
・リセット
　ツールバーの登録を出荷時の状態にします。

ツールバーはモニタ起動時に、前回のモニタ終了時のツールバーの状態を回復します。

・モードコマンド
　モニタの動作モードとして、ＣＰＵの動作モードとクロック、ソースウィンドウでソースプログラム表示時のタブサイズ、ユーザプログラムのソースファイルの存在するデレクトリ、ステップ実行の単位、コマンドパラメータにシンボルを使用するときに%記号を付加するかしないかを設定できます。
（「SET コマンド」、「ICESRC コマンド」、「CAL コマンド」参照）

4.2.1 ビルド機能

モニタは、ユーザプログラムを構成するソースファイル群、ヘッダファイル等との依存関係、コンパイル、アセンブル、リンケージエディタ実行時の各オプションをまとめてプロジェクトとして管理します。
プロジェクトはプロジェクトファイル（拡張子名ＭＡＫ）で管理されます。

(1) 新規作成 コマンド

新規にプロジェクトを作成しプロジェクトをロードます。
本コマンドを選択しますと［新規プロジェクトファイルを開く］のダイアログボックスがオープンします。［ファイル名］に拡張子名がＭＡＫのプロジェクトファイル名をキ－入力して［開く］ボタンを選択します。

次に［プロジェクトへ追加する］のダイアログボックスがオープンします。ユーザプログラムを構成するソースファイルを選択してください。マウスをクリックしますと１ファイルずつ選択します。コントロールキーを押しながらクリックしますと複数のファイルを選択できます。シフトキーを押しながらクリックしますと連続してファイルを選択できます。ファイルの選択後［ＯＫ］ボタンを選択しますとプロジェクトにソースファイルが追加され、プロジェクトをロードします。
プロジェクトロード後は、［編集］コマンドでもプロジェクトにファイルを選択することができます。

拡張子名がＣで始まるファイルはＣコンパイラのソースファイルとして扱います。それ以外のファイルはアセンブラのソースファイルとして扱います。

(2) 開く

作成済みのプロジェクトファイルをロードします。
本コマンドを選択しますと［プロジェクトファイルを開く］のダイアログボックスがオープンします。拡張子名がＭＡＫのプロジェクトファイル名を選択して［開く］ボタンを選択してプロジェクトをロードします。

(3) 編集

ロ－ドされているプロジェクトにソースファイルの追加、削除を行います。
本コマンドを選択しますと［プロジェクトの編集］のドキュメントウィンドウがツリ－ビュ－形式でオープンします。

プロジェクトにソースファイルを追加するときは、マウスの右ボタンをクリックしますとポップアップメニューがオープンしますので［ファイルの追加］のコマンドを選択します。［プロジェクトへ追加する］のダイアログボックスがオープンしますので、上記「新規作成」に記述されている手順に従いソースファイルを追加してください。

プロジェクトからソースファイルを削除するときは、削除したいソースファイルをマウスの左ボタンで選択後、Ｄｅｌｅｔｅキーを押します。または、マウスの右ボタンをクリックして、ポップアップメニューがオ－プン後、［ファイルの削除］のコマンドを選択します。選択されたソースファイルが削除されます。

プロジェクトのソースファイルをダブルクリックしますとそのソースファイルをソースウィンドウに表示します。

(4) オプション設定

Ｃコンパイラ、アセンブラ、リンカ、のオプションと環境設定を行います。
本コマンドを選択しますと［Ｃコンパイラ］、［アセンブラ］、［リンカ］、［ロード］、［環境設定］のタグコントロールがオープンします。タグコントロールを選択することによりオプションを設定できます。次に、各タグコントロールとオプションの意味を下記に示します（Ｃコンパイアセンブラ、リンケージエディタのオプションの詳細な意味に関しては各マニュアルを参照してください）。

(a) Ｃコンパイラ

デバッグ情報を出力するかしないか、および、最適化するかしないかをラジオボタンで選択できます。対象ＣＰＵの種別をコンボボックスで選択できます。［インクルードパス］でインクルードファイルの取り込み先パス名を指定します。複数のインクルードパスを指定するときはカンマで区切ってください。

例） J:\W_ICE　　　　　　　　　　（一つのインクルードパスを指定）
　　 J:\W_ICE,J:\W_SIM　　　　　　（二つのインクルードパスを指定）

［その他］でコンパイラに対して他のオプションを指定できます。複数のオプションを指定するときは空白で区切ってください。

例） -section=p=PROG　　　　　　　（一つのオプションを指定）
　　 -section=p=PROG -define=DEF1　（二つのオプションを指定）

(b) アセンブラ

デバッグ情報を出力するかしないか、および、最適化するかしないかをラジオボタンで選択できます。
対象ＣＰＵの種別をコンボボックスで選択できます。［インクルードパス］でインクルードファイルの取り込み先パス名を指定します。複数のインクルードパスを指定するときはカンマで区切ってください。

例） J:\W_ICE　　　　　　　　　　（一つのインクルードパスを指定）
　　 J:\W_ICE,J:\W_SIM　　　　　　（二つのインクルードパスを指定）

［その他］でアセンブラに対して他のオプションを指定できます。複数のオプションを指定するときは空白で区切ってください。

例） -list=c:\test\asm.lst　　　　　（一つのオプションを指定）
　　 -list=c:\test\asm.lst -section　（二つのオプションを指定）

(c) リンカ

次のオプションに関して指定できます。

［セクション］ではセクションの先頭アドレスのパラメータを指定します（リンカの「ＳＴＡＲＴオプション」）。

例） P(1000),C(0D000)　　　　（セクションＰを１０００番地から、セクションＣをＤ０００番地から割り付けます。）

［ＲＯＭ化］ではＲＯＭ化支援機能のパラメータを指定します（リンカの「ＲＯＭ化オプション」）。

例）「ＲＯＭ化」のオプションは「セクション」のオプションと組み合わせて使用します。

[ｾｸｼｮﾝ]　P,D(400),R,B(10000)　（セクションＰ，Ｄを４００番地から、セクションＲ，Ｂを１００００番地から割り付けます。）
[ROM化]　(D,R)　　　　　　　　（セクションＤと同じ大きさのセクションＲを確保します。セクションＲの割り付け番地は上記で指定します。）

［実行アドレス］では実行開始アドレスを外部定義シンボルで指定します（リンカの「ＥＮＴＲＹオプシュン」）モニタはロード直後指定されたアドレスにプログラムカウンタを設定します。

例） _main　　　　　　　（ロード直後、Ｃソースのｍａｉｎ関数にプログラムカウンタを設定します。）

［その他］ではその他のオプションを指定します。

例）-define=SYM1(10) -define=SYM2(20) （未定義外部参照シンボルを定義します。）

(d) ロード

ビルド実行後に、ユーザプログラムのロードを実行するかどうかをラジオボタンで選択します。

(e) 環境設定

コンパイラ、アセンブラ、リンケージエディタ、ＩＣＥＳＹＭの実行ファイルが存在するパス名等を指定します。それぞれ、参照用のボタンが用意されています。

・［ディレクトリ指定］

コンパイラ　：　コンパイラの実行ファイル群が存在するディレクトリを絶対ディレクトリ名で指定します。

アセンブラ　：　アセンブラの実行ファイルが存在するディレクトリを絶対ディレクトリ名で指定します。

リンカ　：　リンケージエディタの実行ファイルが存在するディレクトリを絶対ディレクトリ名で指定します。

ＩＣＥＳＹＭ　：　本モニタと共に提供されます、ＩＣＥＳＹＭの実行ファイルが存在するディレクトリを絶対ディレクトリ名で指定します。

・［メッセージ出力先］

ロギングファイル名：コンパイラ、アセンブラ、リンカのメッセージをファイルに保存したい場合、絶対ディレクトリでファイル名を指定します。

(5) 閉じる

プロジェクトをクローズします。プロジェクトを新たにロードしない限りビルド等は実行できません（プロジェクトロード後、ロードコマンドを実行してもファイルメニューにロードモジュール名は挿入されません。プロジェクトをクローズ後、ロードコマンドを実行しますとロードモジュール名が挿入されます）。

(6) コンパイル

コンパイルを実行するソースウィンドウまたはテキストウィンドウを選択後、本コマンドを選択しますとコンパイルを実行します。コインパイルオプションはオプション設定で指定したオプションが使用されます。

(7) アセンブル

アセンブルを実行するソースウィンドウまたはテキストウィンドウを選択後、本コマンドを選択しますとアセンブルを実行します。アセンブルオプションはオプション設定で指定したオプションが使用されます。

(8) ビルド

変更のあったソースファイルのコンパイル、アセンブルを実行してロードモジュールを作成します。作成手順は次の通りです。

オブジェクトファイルの作成時刻と、そのオブジェクトを構成するソースファイル（ダブルコーテーションで囲まれたインクルードファイルを含みます）の作成時刻を比較して、ソースファイルの時刻の方が新しかったとき、コンパイル、アセンブルを実行して新たにロードモジュールを作成します。

インクルードファイルは、ソースファイルのあるディレクトリ、コンパイル、アセンブルオプションで指定されたディレクトリの順に検索されます。

(9) リビルド

全ソースファイルをコンパイル、アセンブルしてロードモジュールを作成します。

(10)ビルドの中止

ビルド、リビルド実行中に本コマンドを選択しますとビルドを中断します。中断は、コンパイル、アセンブル等の終了時に行われます。

(11)依存関係の調査

ソースファイルで使用するインクルードファイル（Cコンパイラでは #include アセンブラでは、.include 文）に変更があった場合、依存関係を更新する必要があります。

このような場合、変更のあったソースファイルを表示しているソースウィンドウまたはテキストウィンドウを選択後、本コマンドを選択しますと、依存関係を調査してビルド実行時に反映させることができます。

(12)全ての依存関係の調査

プロジェクトを構成する全てのソースファイルに対して依存関係を調査してビルドに反映します。

4.3 ツールバー

ツールバーには良く使用されるメニューバーのコマンドが登録されています。
ツールバーのボタンにマウスカーソルを置きますとツールヒントが表示されます。
［セットアップ］－［カスタマイズ］でツールバーのコマンドを変更できます。
表４－２にツールバーのコマンド一覧をあげます。

表４－２　ツールバー

ツールバーのコマンド名	機能
ＬＯＡＤ	ユーザプログラムのロード
ＲＥＳＥＴ	レジスタの初期化
ＧＯ	ユーザプログラム（Go）の実行
ＧＯＷ	ユーザプログラムの（Gow）実行
ＧＯＲＥＳ	ユーザプログラムのリセットスタート
ＳＴＥＰ	ステップ実行
ＰＳＴＥＰ	関数ステップ実行
ＲＥＴＵＲＮ	関数の呼び出し元への復帰
ＳＥＴ　ＭＯＤＥ	動作モード設定
ＭＡＰ	マップウィンドウの表示
ＳＯＵＲＣＥ	ソースウィンドウの表示
ＭＩＸ	混在モードウィンドウの表示
ＯＰＥＮ	テキストウィンドウの表示
ＣＯＭＭＡＮＤ	コマンドウィンドウの表示
ＩＮＳＰＥＣＴ	インスペクトウィンドウの表示
ＷＡＴＣＨ	ウオッチウィンドウの表示
ＲＥＧＳ	レジスタウィンドウの表示
ＤＵＭＰ	ダンプウィンドウの表示
Ｉ／Ｏレジスタの表示	Ｉ／Ｏレジスタウィンドウの表示
メモリツリーの表示	メモリツリーウィンドウの表示
ＴＤＭＰ	トレースダンプの表示
ＣＰＵ状態の表示	ＣＰＵステータスウィンドウの表示
ＬＯＣＡＬ	ローカルウィンドウの表示
ＢＡＣＫ	バックトレースウィンドウの表示
スタックの表示	スタックウィンドウの表示
ＣＡＬ	数式計算ウィンドウの表示
？ＳＹＭ	シンボルウィンドウの表示
ＢＵＩＬＤ	ユーザプログラム更新
ＲＥＢＵＩＬＤ	ユーザプログラム更新（全ｿｰｽﾌｧｲﾙｺﾝﾊﾟｲﾙ）
ＣＯＭＰＩＬＥ	コンパイル、アセンブル実行
ＳＴＯＰ	エミュレーションの中止

注）ツールバーの［ＳＴＯＰ］はＧＯＷ、ＰＳＴＥＰ、ＣＯＭＥコマンドを実行してブレークが成立せずにコマンド待ちにならないときにも、ユーザプログラムの実行の中断が行えます。

4.4 ドキュメントウィンドウ

4.4.1 ソースウィンドウ

ソースウィンドウはユーザプログラムのソースを表示します。（コンパイル、アセンブル、リンク時にDEBUG オプションが必要です。

ソースウィンドウには縦、横のスクロールバーが付加されます。

ソースウィンドウがアクティブなときに、上向き矢印キー、下向き矢印キー、Page Up 、Page Down 、（PC9801では、ROLL UP キー、ROLL DOWN キー）によるスクロールが可能です。

現在のＰＣの位置が黄色い矢印で表示されます。

ソースウィンドウは、ポップアップメニュー、インスペクト機能をサポートします。

(1) ポップアップメニュー

図４－３にソースウィンドウの機能を示します。

対応するソース行に命令があるときは先頭に四角の枠が表示されます。
Ｓの記号は、該当する行にソフトブレークが設定されていることを表します。
Ｈの記号は、該当する行にＰＢブレークが設定されていることを表します。
枠内をダブルクリックすることにより、ソフトブレークの設定、解除が可能です。

```
 [S]  00022        switch (i) {
      00023         case  0:
 [H]                      sub_0 (p);
          COME
          ｿﾌﾄﾌﾞﾚｰｸ設定
          PC ﾌﾞﾚｰｸ設定
          SET PC
          FIND
          JUMP
          COPY
          混在表示
```

先頭に四角の枠が表示されている行をマウスカーソルで選択して、マウスの右ボタンをクリックしますと現在実行できるポップアップメニューが表示されます。表４－３にポップアップメニューのコマンドと機能を示します。コマンドの選択方法は、右ボタンを押したままカーソルを移動させ、実行したいコマンドの位置で右ボタンを離します。

図４－３　ソースウィンドウの機能

表４－３　ソースウィンドウのポップアップメニュー

ポップアップメニューのコマンド	機能
COME	指定したソース行まで実行します。（「COME コマンド」参照）
ｿﾌﾄﾌﾞﾚｰｸ設定	指定したソース行にソフトブレークを設定します。すでにソフトブレークが設定されているときは解除します。（「AT コマンド」参照）
PC ﾌﾞﾚｰｸ設定	指定したソース行にＰＣブレークを設定します。すでにＰＣブレークが設定されているときは解除します。（「PB コマンド」参照）
SET PC	指定したソース行にＰＣを設定します。
FIND	ソースファイル内の文字列を検索します。
JUMP	ダイアログボックスで指定した行番号のソース行を表示します。
COPY	ソースウィンドウ上でマウスの左ボタンを押しながらある範囲を選択したときに（背景色が黒になります）、選択された文字列をクリップボードにコピーします。
混在表示	指定したソース行に対応する混在モードウィンドウを表示します。

(2) インスペクト機能

変数、または関数の位置にマウスカーソルを移動して静止させますと、変数の内容を一行で表示するチップインスペクト機能をサポートしています。また、変数をダブルクリックしますと、インスペクトウィンドウがオープンします。関数の位置でダブルクリックしますと、関数定義のあるソースウィンドウをオープンし該当行を表示します。表示内容に関しましては、「インスペクトウィンドウ」を参照してください。

4.4.2 混在モードウィンドウ

混在モードウィンドウはユーザプログラムのソース、またはシンボルと機械語の対応を表示します。
混在モードウィンドウには縦、横のスクロールバーが付加されます。
混在モードウィンドウがアクティブなときに、上向き矢印キー、下向き矢印キー、Page Up 、Page Down 、(PC9801では、ROLL UP キー、ROLL DOWN キー）によるスクロールが可能です。
現在のＰＣの位置が黄色い矢印で表示されます。
図４－４に混在モードウィンドウの機能を示します。

対応する機械語に命令があるときは先頭に四角の枠が表示されます。
Ｓの記号は、該当する行にソフトブレークが設定されていることを表します。
Hの記号は、該当する行にPCブレークが設定されていることを表します。
枠内または枠が存在する行をダブルクリックすることにより、ソフトブレークの設定、解除が可能です。

```
      ＵＳＥＲ２．Ｃ：０００９          ｍａｉｎ()
[S]   ００００００００  ４Ｆ  ２２      ＳＴＳ．Ｌ   ．．．
[H]   ０００００００２  ７Ｆ  Ｃ８      ＡＤＤ       ．．．
      ＵＳＥＲ２．Ｃ：００１５          ｃｏｕ＿０=．．．
[ ]   ０００００００４  Ｅ３  ００      ＭＯＶ       ．．．
[ ]   ０００００００６                  ＭＯＶ．Ｌ   ．．．
                  ＣＯＭＥ
                  ｿﾌﾄﾌﾞﾚｰｸ設定
                  PB ﾌﾞﾚｰｸ設定
                  SET PC
                  ＪＵＭＰ
```

先頭に四角の枠が表示されている行をマウスカーソルで選択して、マウスの右ボタンをクリックしますと現在実行できるポップアップメニューが表示されます。表4-4にポップアップメニューのコマンドと機能を示します。コマンドの選択方法は、右ボタンを押したままカーソルを移動させ、実行したいコマンドの位置で右ボタンを離します。

図４－４　混在モードウィンドウの機能

表４－４　混在モードウィンドウのポップアップメニュー

ポップアップメニューのコマンド	機能
ＣＯＭＥ	指定したソース行まで実行します。 （「COME コマンド」参照）
ｿﾌﾄﾌﾞﾚｰｸ設定	指定したソース行にソフトブレークを設定します。 すでにブレークが設定されているときは解除します。 （「AT コマンド」参照）
PC ﾌﾞﾚｰｸ設定	指定したソース行にＰＣブレークを設定します。 すでにＰＣブレークが設定されているときは解除します。 （「PB コマンド」参照）
SET PC	指定したソース行にＰＣを設定します。
ＪＵＭＰ	指定したアドレスから再描写します。 （垂直スクロールバーのサムボタンを選択しても同じ）

4.4.3　コマンドウィンドウ

　コマンドウィンドウは、コマンドのキー入力とその実行結果を表示します。キー入力できるコマンドに関しましては「5　コマンド」を参照してください。メニューバーに登録されていないコマンドも入力可能です。コマンドウィンドウの詳細に関しましては「8 各コマンドの詳細」を参照してください。
　プロンプトは　ｄ＞　（ｄはカレントドライブ）の形式で表示されます。
　他のドキュメントウィンドウがフォーカスを持っているときも、文字、ファンクション、上向き、下向きの矢印キーを入力しますとコマンドウィンドウにフォーカスが移動し、直ちにコマンドを入力することが可能です。

4.4.4　テキストウィンドウ

　テキストウィンドウは、指定されたテキストファイルをオープンします。
　テキストウィンドウには縦、横のスクロールバーが付加されます。

4.4.5　ダンプウィンドウ

　ダンプウィンドウは、指定されたアドレスの内容をダンプ形式で表示します。［ADDRESS］にダンプしたいアドレスを設定し、次に［Draw］ボタンをクリックしますとメモリの内容が表示されます。［Prev］ボタンをクリックしますと、前のページ（アドレスの少ない方）のダンプ、［Next］ボタンをクリックしますと後のページ（アドレスの大きい方）のダンプを実行します。BYTE,WORD,LONG,のラジオボタンでバイト単位、ワード単位、ロングワード単位でのダンプを選択できます。

　マウスカーソルを１６進数の表示部分に移動して数字を入力しますとメモリ内容の書き換えを行います。また、１６進数の表示部分をダブルクリックしますとダイアログウィンドウがオープンしてメモリ内容の書き換えがおこなえます。（式を使用できます。）

　縦のスクロールバーの移動により表示位置を変更できます。
　ダンプウィンドウのサイズを変更しますと、メモリ表示内容ウィンドウのサイズも変更されます。
　ダンプウィンドウは１０個までオープンできます。

4.4.6　ウオッチウィンドウ

　ウオッチウィンドウはある変数の値の変化を表示するウィンドウです。

　ウオッチウィンドウへの変数の登録には、［登録］ボタンを選択して［変数選択ダイアログボックス］をオープンします。数値でアドレスを入力する場合は、［アドレス］に数値を入力して［登録］を選択します。シンボル名で登録する場合はリストボックス内に登録されているシンボル名を選択して［登録］を選択します。［表示選択］を選択しますと、アルファベット順による表示シンボル名の指定が行えます。

　変数の型は、バイト（BYTE）、ワード（WORD）、ロングワード（LONG）、文字列（ASCII）、単精度浮動小数点（FLOAT）、倍精度浮動小数点（DOUBLE）が選択できます。変数の登録個数を［個数］で設定します。また、内蔵レジスタや外部のメモリ空間をウオッチするために［エミュレーションを一時中断］も選択できます。（「WATCH コマンド」参照）
　ウオッチ変数は最大２０個の変数まで登録できます。

　ウオッチウィンドウの変数の変更には、登録されている変数をクリックしてから［変更］ボタンを選択します。
　ウオッチウィンドウから変数の削除には、登録されている変数をクリックしてから［削除］ボタンを選択します。

4.4.7　ローカルウィンドウ

　Ｃ言語でコーデングされたユーザプログラムにおいて、現在のＰＣのある関数のローカル変数の値を表示します。

4.4.8　バックトレースウィンドウ

　Ｃ言語でコーデングされたユーザプログラムにおいて、現在のＰＣのある関数がどの関数から呼ばれているかを表示します。

4.4.9　ＣＰＵステータスウィンドウ

現在のターゲットＣＰＵの状態を表示するウィンドウです。
表示される意味は次のとおりです。

・ＣＰＵ
ターゲットＣＰＵの種類を示します。

・ＥＲＡＭ
内蔵されているエミュレーションＲＡＭを表示します。
ＥＲＡＭ＝２Ｍ　　エミュレーションＲＡＭは２ＭＢ
ＥＲＡＭ＝４Ｍ　　エミュレーションＲＡＭは４ＭＢ

・ＭＯＤＥ
モードを示します。
（）内はユーザケーブルによるモードを示します。

・ＣＬＯＣＫ
ターゲットＣＰＵが現在ユーザクロックかまたはＩＣＥが貸し出すシステムクロックで動作しているかを示します。

・ＡＤＤＲ
ターゲットＣＰＵ上でユーザプログラムが実行中であるとき、アドレスバスの値を示します。

・ＰＲＯＢ
外部プローブの値を示します。
ＴＲＣコマンドでＰＲＯＢＥパラメータが指定されているときに表示します。

・ＬＩＮＥ
ＣＰＵの端子の値を表示します。ＨＩＧＨ（’１’）のとき通常の色で表示し、ＬＯＷ（’０’）のとき赤色の反転で表示します。
表示する端子は次のとおりです。

ＲＥＳ	リセット端子
ＮＭＩ	ノンマスカブル割込み端子
ＳＴＢＹ	スタンバイ端子
ＷＡＩＴ	ウェイト端子
ＢＲＥＱ	バス権要求端子

・ＳＴＡＴＵＳ
ターゲットＣＰＵの状態を表示します。
表示する状態は次のとおりです。

ＤＭＡ	ＤＭＡサイクル
ＲＥＦ	リフレッシュサイクル
ＰＦ	プリフェッチサイクル
ＳＴＢＹ	スタンバイ状態
ＳＬＥＥＰ	スリープ状態
ＢＲＥＬ	バスリリース状態

4.4.10　レジスタウィンドウ

レジスタの値を表示します。
レジスタの値を変更するときはレジスタ名のボタンを選択します。レジスタ値の変更のダイアログボックスがオープンします。ここで、レジスタの値を設定して［ＯＫ］を選択しますと値が変更できます。［Ｃａｎｃｅｌ］を選択しますと変更の中止、［Ｒｅｓｅｔ］で元の値に戻ります。

4.4.11　スタックウィンドウ

現在スタックに積まれている内容を表示します。

4.4.12　シンボルウィンドウ

　現在ロードされているシンボルを表示します。
［表示選択］を選択しますと、アルファベット順による表示シンボル名の指定が行えます。

4.4.13　数式計算ウィンドウ

　数式の計算を行うウィンドウです。使用できる式に関しましては「5.2　式」を参照してください。

　計算結果は、符号なし十進数（DECIMAL(U)）、符号付き十進数（DECIMAL(S)）、十六進数（HEXA）、二進数（BINARY）、文字列（ASCII）で表示します。

4.4.14　マップウィンドウ

　マッピングの状態を表示します。
　［中止］を選択しますとウィンドウをクローズします。［全解除］でモニタ起動直後のマッピングの状態に戻します。

　マッピングを変更したいときは、変更したい行をクリックして［変更］を選択しますと、マップの変更のダイアログボックスがオープンします。ここで、開始アドレス、終了アドレス、モードを設定します。（使用できるﾓｰﾄﾞ、意味に関しましては「MAP コマンド」を参照してください。）［ＯＫ］ボタンを選択しますと、設定したマッピングに変更されダイアログボックスはクローズします。（設定に誤りがあるときはクローズしません。）
［キャンセル］の選択で変更の中止、［リセット］の選択で元のマッピングの状態に戻します。

4.4.15 トレースダンプウィンドウ

現在取得されているトレース内容を表示します。
図４－５にトレースダンプウィンドウの機能を示します。
トレースダンプのメニュー、または、ツールバーのトレースダンプボタンを選択しますと描写を行います。
トレースダンプウィンドウがアクティブなときに、上向き矢印キー、下向き矢印キー、Page Up 、Page Down（PC9801では、ROLL UP キー、ROLL DOWN キー）によるスクロールが可能です。

```
f-no    ton   addr    data    ma     st   iack   7654321
4051          00000D   55    IROM    PF          0000000
4050          0FFCDA  00     IROM    W           0000000
     $  USER2.C:0012          count = 0 ;
     *  00000C   19 55               SUB.W       R5,R5
                                                          ) ;
                    Ｆｉｎｄ
                    Ｓｅｌｅｃｔ
```

マウスのボタンをクリックしますとポップアップメニューが表示されます。
表４－５にポップアップメニューのコマンドと機能を示します。

図　４－５　トレースダンプウィンドウの機能

表　４－５　トレースダンプウィンドウのポップアップメニュー

ポップアップメニュー	機能
Ｆｉｎｄ	トレースウィンドウ内の文字列を検索します。
Ｓｅｌｅｃｔ	次のトレースウィンドウの表示範囲や条件を指定できます。 表示を開始するフレーム番号と表示する最大の行数（初期値は１０００行）。指定した文字列を含む行を選択して表示。

4.4.16 カバレージウィンドウ

カバレージの測定結果を表示します。通過範囲を黒の線で表します。
表示範囲は、８Ｋバイトと６４Ｋを選択できます。表示開始アドレスを指定できます。

4.4.17 パフォーマンスウィンドウ

モジュールの実行時間、実行回数の測定結果、および、実行時間の分布測定の結果を表示します。
（「PER コマンド」参照）

4.4.18 インスペクトウィンドウ

インスペクトウィンドウはＣ言語の構造体変数およびポインタ・配列変数の内容を表示するウィンドウです。

インスペクトウィンドウへの変数の登録は、メニューバーの「データ(D)」の［インスペクト(I)］を選択して［変数選択ダイアログボックス］をオープンします。
次にリストボックス内に登録されているシンボル名を選択して［インスペクト］ボタンをクリックします。

変数選択ダイアログボックスには関数内での有効なローカル変数も表示されます。
この場合、右側のアドレス・レジスタ名の前に'L'が付きます。
INSPECTコマンドを使用することによりコマンドウィンドで登録することも可能です。
インスペクトウィンドウは最大５０個まで表示できます。

タイトルバーの下のボタンの機能を表４－６に示します。

表４－６　ボタンの機能

ボタン名	機　　能
Repaint	変数の再表示
Expand	変数の展開(構造体メンバ・配列・ポインタ)
Range	配列・ポインタの登録範囲の変更
Change	変数値の変更
Close	インスペクトウィンドウのクローズ

変数の展開(Expand)は、表示されているメンバ名または配列要素を直接ダブルクリックしても可能です。

情報が多い場合にはリストボックス内に１００行まで登録されます。
他の部分を参照する場合には［Range］を選択して登録範囲を変更します。
登録範囲の変更は負数も可能です(例:TOP=-4,END=7)。
また、TOP=0、END=0を指定した場合元の範囲に戻ります。

4.4.19 Ｉ／Ｏレジスタウィンドウ

内蔵レジスタの値をツリービュー形式で表示します。
親項目にモジュール名が、子項目に内蔵レジスタ名、アドレス、内蔵レジスタの値が表示されます。
子項目をダブルクリックしますとダイアログボックスがオープンして内蔵レジスタの値を変更できます。
マウスの右ボタンをクリックしますと、エミュレーションを一時中断して内蔵レジスタの値を表示するかどうかの選択がおこなえます。

ドラッグアンドドロップにより表示位置を変更できます。

本ウィンドウをクローズして再びオープンしますと表示位置は初期状態に戻ります。

内蔵レジスタの定義はＩＯＲＥＧ．ＤＡＴのファイルで行います。
ＩＯＲＥＧ．ＤＡＴのフォーマット形式は次の通りです。

［＜モジュール名＞］
＜レジスタ名＞　＜アドレス＞　＜サイズ＞
＜レジスタ名＞　＜アドレス＞　＜サイズ＞
・・・・・・・・・・・・・・・・

＜モジュール名＞　："["と"]"で囲むことによりモジュール名を指定します。
＜レジスタ名＞　　：内蔵レジスタ名を指定します。
＜アドレス＞　　　：内蔵レジスタのアドレスです。
＜サイズ＞　　　　：内蔵レジスタをリード／ライトするときのサイズと表示で下記のいずれかを指定します。
Ｂ・・・バイトでアクセスします。
Ｗ・・・ワード（２バイト）でアクセスします。
Ｌ・・・ロングワード（４バイト）でアクセスします。
（Ｂ、Ｗ、Ｌを小文字のｂ、ｗ、ｌで記述したときは、ライトのみ可能なレジスタとしてデータの表示はおこないません。）
１・・・１バイトの文字
２・・・２バイトの文字
４・・・４バイトの文字
８・・・８バイトの文字
＃、；　　　　　　：＃（シャープ）、；（セミコロン）以降はコメントとして扱われます。

具体的な例は提供されますＩＯＲＥＧ．ＤＡＴを参照してください。

補注）Ｉ／Ｏレジスタウィンドウの”ｔｅｓｔモジュール”には、よく参照するレジスタを登録しますと参照が容易になります。登録方法は、レジスタをドラッグアンドドロップで登録する方法と、ＩＯＲＥＧ．ＤＡＴに予め登録しておく方法があります。

4.4.20 メモリツリーウィンドウ

メモリツリーウィンドウはＩ／Ｏレジスタウィンドウと同様の機能を持ちます。

メモリの値をツリービュー形式で表示します。
親項目にモジュール名が、子項目に変数名、アドレス、変数の値、または、シンボル名（先頭に%が付加されます）と、シンボルの値が表示されます。
子項目をダブルクリックしますとダイアログボックスがオープンして変数の値を変更できます。
マウスの右ボタンをクリックしますと［エミュレーションを一時中断する］と［追加する］のコマンドが選択できます。［エミュレーションを一時中断する］を選択しますと、エミュレーションを一時中断して変数の値を表示します。
［追加する］を選択しますとメモリツリ－ウィンドウに表示する変数を追加、挿入できます。挿入される位置は、コマンド指定前に選択された親項目になります。親項目が選択されていない場合は新たに［追加］の親項目が作成され変数が追加されます。
［追加する］ボタンを選択しますと［変数選択ダイアログボックス］がオープンします。数値でアドレスを入力する場合は、［アドレス］に数値を入力して［登録］を選択します。シンボル名で登録する場合はリストボックス内に登録されているシンボル名を選択して［登録］を選択します。［表示選択］を選択しますと、アルファベット順による表示シンボル名の指定が行えます。
変数の型は、バイト（BYTE）、ワード（WORD）、ロングワード（LONG）、文字（ASCII 1、2、4、8文字）が選択できます。外部のメモリ空間をウオッチするために［エミュレーションを一時中断］も選択できます。
ドラッグアンドドロップにより表示位置を変更できます。
本ウィンドウをクローズして再びオープンしますと表示位置は初期状態に戻ります。
メモリツリーの定義はＭＥＭＴＲＥＥ．ＤＡＴのファイルで行い、ユーザが定義する必要があります。

ＭＥＭＴＲＥＥ．ＤＡＴのフォーマット形式は次の通りです。

［＜モジュール名＞］
＜変数名＞　　　＜アドレス＞　＜サイズ＞
%＜シンボル名＞　　　　　　　＜サイズ＞
・・・・・・・・・・・・・・・・・・・

＜モジュール名＞　："["と"]"で囲むことによりモジュール名を指定します。
＜変数名＞　　　　：変数名を指定します。（任意の文字列です）
＜アドレス＞　　　：変数のアドレスを指定します。
%＜シンボル名＞　：変数としてシンボル名を指定するときは先頭に%を付加します。
　　　　　　　　　　アドレスの指定は不要です。
＜サイズ＞　　　　：変数をリード／ライトするときのサイズで下記のいずれかを
　　　　　　　　　　指定します。
　　　　　　　　　　　Ｂ・・・バイトでアクセスします。
　　　　　　　　　　　Ｗ・・・ワード（２バイト）でアクセスします。
　　　　　　　　　　　Ｌ・・・ロングワード（４バイト）でアクセスします。
　　　　　　　　　　　１・・・１バイトの文字
　　　　　　　　　　　２・・・２バイトの文字
　　　　　　　　　　　４・・・４バイトの文字
　　　　　　　　　　　８・・・８バイトの文字

＃、；　　　　　　：＃（シャープ）、；（セミコロン）以降はコメントとして扱われます。

具体的な例は提供されますＭＥＭＴＲＥＥ．ＤＡＴを参照してください。

4.4.21プロジェクトウィンドウ

　ロードモジュールを構成するソースファイルと関数をツリービュー形式で表示します。
　プロジェクトウィンドウには、［関数］と［ソースファイル］のタグコントロールが表示されます。
　［関数］のタグを選択しますと、親項目がソースファイル名、子項目にソースファイルに含まれる関数名の一覧が表示されます。子項目の関数名をダブルクリックしますと該当する関数のソースファイルがソースウィンドウに表示されます。
　［ソースファイル］のタグを選択しますと、ロードモジュールを構成するソースファイル名の一覧が表示されます。ソースファイル名をダブルクリックしますと該当するソースファイルのソースウィンドウが表示されます。

　ドラッグアンドドロップにより表示位置を変更できます。

　本ウィンドウをクローズして再びオープンしますと表示位置は初期状態に戻ります。

５．コマンド

この章では、コマンドウィンドウで使用できるコマンドについで述べます。

5.1 コマンドの入力形式

(1) コマンドの入力形式
モニタのコマンド形式は次のようになっています。

d ＞　コマンド名［／オプション］　　［ パラメータ ］　・・・
　　　　　　　　　　　　　　　　　［ キーワード ／ パラメータ ］　・・・

d	：	カレントドライブです。
＞	：	プロンプト記号です。
[]	：	[] 内はコマンドにより省略可能です。
コマンド名	：	モニタのコマンドの名称です。
オプション	：	コマンド実行時のオプションです。”／”記号で区切る事により複数のオプションが指定できます。
パラメータ	：	コマンド実行時のパラメータです。 パラメータが数値の場合は式を使用することができます。
キーワード	：	ある特定のパラメータを指定するときに使用します。

注意）
- ・コマンドに対して不当なオプションを指定したときは無視されて実行されます。
- ・１つのオプションのみ指定できるコマンドに対して複数のオプションを指定すると一番最初に指定したオプションが有効となります。
- ・パラメータ、キーワードは空白で区切ります。
- ・パラメータに、；、＜、＞、｜、の文字を使用するときは、”（ダブルコーテーション記号）で囲んでください。

(2) 複数コマンドの指定方法
複数のコマンドを指定する時に、；で区切ります。

例）　ＤＵＭＰ　１０００　１０１０　；　ＧＯ
　　　　ＤＵＭＰコマンド実行後、ＧＯコマンドを、実行します。

コマンドラインは最大１２８文字まで入力可能です。

5.2 式

式は、数値定数、レジスタ定数、間接参照、シンボルから成り、これらを演算子で結合したものです。
演算は全て、３２ビットで行われます。
式は、コマンドパラメータ内の全ての数値を表すパラメータに使用できます。

(1) 数値定数
数値定数は、２進、８進、１０進、および１６進が使用できます。
基数が使用された事を示す文字とそれに続く数字で表現されます。
基数が省略されたときは、１６進数とみなしますが、パスカウントなどの回数を指定するときは１０進数とみなします。

Ｂ’Ｂ＃	−−−−	２進数
Ｏ’Ｏ＃	−−−−	８進数
Ｄ’Ｄ＃	−−−−	１０進数
Ｈ’Ｈ＃	−−−−	１６進数

例）

Ｂ’１１００	Ｂ＃１１００	（２進数の１１００）
Ｄ’９８７６	Ｄ＃９８７６	（１０進数の９８７６）
Ｈ’ＦＥＤＣ	Ｈ＃ＦＥＤＣ	（１６進数のＦＥＤＣ）

(2) レジスタ定数

現在のレジスタの値をレジスタ定数として使用でき、レジスタ名称を指定して表現します。
次のレジスタ定数が使用できます。

．ＥＲ０、．ＥＲ１、．ＥＲ２、．ＥＲ３、．ＥＲ４、．ＥＲ５、．ＥＲ６、．ＥＲ７
．ＰＣ、．ＣＣＲ
．Ｒ０、．Ｒ１、．Ｒ２、．Ｒ３、．Ｒ４、．Ｒ５、．Ｒ６、．Ｒ７

(3) 間接参照

レジスタの値や、メモリの内容をポインタとする間接参照が可能です。

Ｂ＊番地	－－－－	指定された番地（数値）の内容１バイトの値。
Ｗ＊番地	－－－－	指定された番地（数値）の内容２バイトの値。
Ｌ＊番地	－－－－	指定された番地（数値）の内容４バイトの値。

例）
＞ＤＵＭＰ Ｗ＊．Ｒ０
レジスタＥＲ０ の値の番地の内容が指すメモリ内容をダンプします。

(4) シンボル

ＬＯＡＤコマンドで入力したシンボルの値を使用できます。
コマンドのパラメータにシンボルを指定するときに、シンボル名の前に%を付加するか、しないかを指定できます。指定の方法は、メニューバーの［セットアップ］－［モード］、ツールバーの［Ｍｏｄｅ］、またはＣＡＬコマンドで行います。（詳細は「CAL コマンド」参照）

シンボルの形式は次のようになります。

［%］［＜ファイル名＞：］{関数名｜変数名}

[] 内は省略可能です。
{} 内は、その内の一つを選択します。

＜ファイル名＞は、グローバル変数名と関数名、および自動変数名に対しては不要です。
スタティック変数名と関数名に対してファイル名を指定してください。
次に、具体的な例を挙げて説明します。

dump	%g_val	グローバル変数 g_val の表示
dasm	%main	グローバル関数 main の逆アセンブル表示
at	%g_func	グローバル関数 g_func にソフトブレーク設定
dump	%file.c:s_val	ソースファイル file のスタテック変数 s_val の表示
dasm	%file.c:s_func	ソースファイル file のスタテック関数 func の逆アセンブル表示

アセンブラで使用するシンボルは、外部定義の宣言（.ＥＸＰＯＲＴ、.ＧＬＯＢＡＬ制御命令を使用）を行なったものは、グローバル変数となり、他のものはローカル変数となります。

(5) 演算

上記の値にたいして、下記の演算が行えます。

＋	加算
－	減算
＊	乗算
／	除算
&	ビットごとのＡＮＤ
^	ビットごとの排他的ＯＲ
！	ビットごとのＯＲ

(6) 括弧
数式を括弧で囲むことができます。
例）
（. Ｒ０＋１）＊１０

(7) 優先順位
演算子は下記のような優先度を持ちます。

〈高い優先度〉
()
Ｂ＊，Ｗ＊，Ｌ＊
＊，／
＋，－
&
^
！
〈低い優先度〉

5.3 数値の指定方法

コマンドのパラメータの中に、数値（アドレス等）を指定する時には、式で指定することが可能です。

例）
現在のプログラムカウンタ＋１０をＥＲ０　レジスタに代入します。
＞ＳＲＥＧ　ＥＲ０／．ＰＣ＋１０

ＥＲ６　レジスタの値＋１００番地にソフトブレークを設定します。
＞ＡＴ　．ＥＲ６＋１００

８８６６番地の内容２バイトを４倍して１０加えた番地の内容をダンプします。
＞ＤＵＭＰ　４＊Ｗ＊８８６６＋１０

5.4 レンジの指定方法

コマンドのパラメータの中で、指定する番地の範囲を、レンジといいます。
レンジの指定方法には次の二つの形式があります。

・先頭番地と最終番地を指定する方法

〈先頭番地〉〈最終番地〉

・先頭番地とサイズを指定する方法

〈先頭番地〉　￥〈サイズ（長さ）〉

例）１０００番地から１０ＦＦ番地までをダンプします。

”〈先頭番地〉〈最終番地〉”の形式では次のようになります。
＞ＤＵＭＰ　１０００　１０ＦＦ

”〈先頭番地〉　￥〈サイズ〉”の形式では次のようになります。
＞ＤＵＭＰ　１０００　￥１００

5.5 コマンドシェルの機能

コマンドシェルはユーザのコマンド入力の操作を容易にします。
コマンドシェルはコマンドの編集機能とヒストリ機能があります。

(1) コマンドの編集機能
入力したコマンドに対して次のような行の修正機能があります。
コマンドの修正した文字は上書きされずに、挿入されます。

上向き矢印	このキーが入力されるごとに、一つ前に実行したコマンドを表示します。
下向き矢印	上向き矢印キーまたは下向き矢印キーで呼びだされたコマンドの次に実行したコマンドを表示します。
右向き矢印	カーソルを一つ右へ移動します。
左向き矢印	カーソルを一つ左へ移動します。
ＢＳ （ＣＴＲＬ－Ｈ）	カーソルの左の位置にある文字を削除します。
ＤＥＬ	カーソルの位置にある文字を削除します。
ＣＴＲＬ－Ｙ ＥＳＣ	現在の入力している行をクリヤします。

(2) コマンドヒストリ機能

コマンドヒストリの機能により前回入力したコマンドを、容易に呼びだすことができます。
コマンド入力の履歴を記憶するヒストリバッファとして１Ｋバイト用意してあります。
下記にヒストリ機能の概要を表わします。詳細については　ＨＩＳ、！、：のコマンドの説明を参照してください。

ＨＩＳ	ヒストリに記憶されているコマンドを表示します。
！	コマンドヒストリに記憶されているコマンドを選択し、実行します。
：	コマンドヒストリに記憶されているコマンドを選択し、編集とコマンド入力が可能となります。

(3) 別名のコマンド定義

モニタのコマンドに対して別のコマンド名を定義できます。この機能により、より使いやすいコマンド名に変更することができます。
別名の定義用バッファとして、２Ｋバイト用意してあります。
下記に機能の概要を表わします。詳細については「ＡＬＩＡＳコマンド」を参照してください。

ＡＬＩＡＳ　別名　文字列
指定した文字列に対して別名の定義を行ないます。
これ以降、コマンド入力で別名の文字列を入力しますと指定された文字列に置き換えて実行します。

初期値はＩＣＥＩＮＩＴ．ＤＡＴのファイルで定義されています（エディタで変更可能です）。

(4) ファンクションキーの定義

ファンクションキーに対して別のコマンド名（文字列）を定義できます。
この機能により、よく使用するコマンドをファンクションキーに登録することができます。
下記に機能の概要を表わします。詳細については「ＦＫコマンド」を参照してください。

ＦＫ　番号　文字列
指定したファンクションキーに対して文字列を割り当てます。

初期値はＩＣＥＩＮＩＴ．ＤＡＴのファイルで定義されています（エディタで変更可能です）。

5.6 入出力先のリダイレクト機能

通常コマンドは、キーボードからコマンドやパラメータを入力して実行し、その結果をコンソールへ出力（表示）します。
これらの入出力先は、次のように入出力先の変更記号（”＞”、”＞＞”、”＜”）によりコマンドの入出力先を変更（リダイレクト）できます。

ＤＵＭＰ １００ ２００ ＞ ＤＵＭＰＦＩＬＥ
ＤＵＭＰＦＩＬＥ の ファイルに出力。

ＤＵＭＰ １００ ２００ ＞＞ ＤＵＭＰＦＩＬＥ
ＤＵＭＰＦＩＬＥ の ファイルに追加出力。

補注）
マクロコマンドに対して入出力先の変更は行えません。

本モニタはコマンドの入出力先のリダイレクト機能により、必要なデータを一時ファイルに退避し、その後再び後でファイルよりデータをリードして設定することができます。

次にこの機能を使用できるコマンドを示します。

・ソフトブレーク条件の設定
全てのソフトブレークの条件をファイルに退避し再び設定します。
>?AT　>　＜ファイル名＞
><　＜ファイル名＞

・ＰＣブレーク条件の設定
ＰＣブレーク　ＰＢ０　の条件をファイルに退避し再び設定します。
>?PB0 >　＜ファイル名＞
><　＜ファイル名＞

全てのPCブレークの条件をファイルに退避し再び設定します。
>?PB >　＜ファイル名＞
><　＜ファイル名＞

・バスブレーク条件の設定
バスブレーク　ＢＢ０　の条件をファイルに退避し再び設定します。
>?BB0　>　＜ファイル名＞
><　＜ファイル名＞

すべてのバスブレークの条件をファイルに退避し再び設定します。
>?BB　>　＜ファイル名＞
><　＜ファイル名＞

・トリガ条件の設定
トリガ条件　ＴＧ１　をファイルに退避し再び設定します。
>?TG1　>　＜ファイル名＞
><　＜ファイル名＞

すべてのトリガの条件をファイルに退避し再び設定します。
>?TG > ＜ファイル名＞
>< ＜ファイル名＞

・外部トリガ出力条件の設定
外部トリガの出力の条件をファイルに退避し再び設定します。
>?EXT > ＜ファイル名＞
>< ＜ファイル名＞

・トレース取得条件の設定
トレース取得の条件をファイルに退避し再び設定します。
>?TRC > ＜ファイル名＞
>< ＜ファイル名＞

・タイマ条件の設定
タイマ（実行時間）の測定条件をファイルに退避し再び設定します。
>?TIM > ＜ファイル名＞
>< ＜ファイル名＞

・外部信号の入力制御
外部信号の入力制御の情報をファイルに退避し再び設定します。
>?LINE > ＜ファイル名＞
>< ＜ファイル名＞

・パフォーマンス測定条件の設定
パフォーマンス測定条件をファイルに退避し再び設定します。
>?PER > ＜ファイル名＞
>< ＜ファイル名＞

・カバレージ測定条件の設定
カバレージ測定条件をファイルに退避し再び設定します。
>?COV > ＜ファイル名＞
>< ＜ファイル名＞

・レジスタ値
全レジスタの値をファイルに退避し再び設定します。
>REG > ＜ファイル名＞
>< ＜ファイル名＞

プログラムカウンタの値をファイルに退避し再び設定します。
>REG PC > ＜ファイル名＞
>< ＜ファイル名＞

・マッピング
マッピング情報をファイルに退避し再び設定します。
>?MAP > ＜ファイル名＞
>< ＜ファイル名＞

・エミュレータ動作環境の設定
エミュレータの動作環境の情報をファイルに退避し再び設定します。
>?SET > ＜ファイル名＞
>< ＜ファイル名＞

・ファンクションキーの設定
ファンクションキーの定義情報をファイルに退避し再び設定します。
>?FK > ＜ファイル名＞
>< ＜ファイル名＞

・サンプリングタイマの設定
サンプリングタイマの情報をファイルに退避し再び設定します。
>?SMPL > ＜ファイル名＞
>< ＜ファイル名＞

5.7 実行および表示の中断

コマンド実行の停止や表示の中断を行ないたい時に次の様なコントロールキーを入力してください。

(1) コマンド実行の強制終了 －－－　ＳＴＯＰキーまたはＣＴＲＬ－ＰＡＵＳＥはＣＴＲＬ－ＰＡＵＳＥ
　現在実行中のコマンドを中断したい時は、ＳＴＯＰキー、（ＰＣ－９８０１の時）、また（ＰＣ／ＡＴの時）を入力してください。
　プログラム実行中にＳＴＯＰまたはＣＴＲＬ－ＰＡＵＳＥ キーを入力しますとユーザプログラムは強制終了されます。

5.8 コマンドの分類

本モニタがサポートしているコマンドと分類を次に示します。

プログラム実行制御

ＣＯＭＥ	指定アドレスまでの実行
ＧＯ	ユーザプログラムの実行
ＧＯＲＥＳ	リセットスタート機能
ＧＯＷ	ブレーク待ちのユーザプログラム実行
ＬＩＮＥ	外部信号の入力制御
？ＬＩＮＥ	外部信号の制御状態の表示
ＰＳＴＥＰ	関数単位の１ステップ実行
ＲＥＴＵＲＮ	呼び出し元関数への復帰
ＳＴＥＰ	ユーザプログラムの１ステップ実行
ＳＴＯＰ	ユーザプログラムの中断

ブレーク条件

ＡＴ	ソフトブレーク条件の設定
？ＡＴ	ソフトブレーク条件の表示
＿ＡＴ	ソフトブレーク条件の削除
ＢＢｎ	バスブレーク条件ｎの設定
？ＢＢｎ	バスブレーク条件ｎの表示
＿ＢＢｎ	バスブレーク条件ｎの削除
ＰＢｎ	ＰＣブレーク条件ｎの設定
？ＰＢｎ	ＰＣブレーク条件ｎの表示
＿ＰＢｎ	ＰＣブレーク条件ｎの削除
ＴＧｎ	トリガ条件ｎの設定
？ＴＧｎ	トリガ条件ｎの表示
＿ＴＧｎ	トリガ条件ｎの解除

トレース制御

ＴＤＭＰ	トレースバッファの表示
ＴＲＣ	トレース条件の設定
？ＴＲＣ	トレース条件の表示
＿ＴＲＣ	トレース条件の解除

タイマ条件

ＴＩＭ	タイマ条件の設定
？ＴＩＭ	タイマ値とタイマ条件の表示
＿ＴＩＭ	タイマ条件の解除

外部トリガ出力条件

ＥＸＴ	外部トリガ出力条件の設定
？ＥＸＴ	外部トリガ出力条件の表示
＿ＥＸＴ	外部トリガ出力条件の削除

パスカウント条件

ＰＡＳＳ	パスカウント計測条件の設定
？ＰＡＳＳ	パスカウント計測条件の表示
＿ＰＡＳＳ	パスカウント計測条件の削除

レジスタオペレーション

ＲＥＧ	レジスタの表示
ＳＲＥＧ	レジスタの設定

メモリオペレーション

ＢＡＣＫ	バックトレースの表示
ＣＨＮＧ	メモリ内容のサーチと変更
ＤＵＭＰ	メモリ内容の表示
ＦＩＬＬ	メモリ内のパターン設定
ＬＯＣＡＬ	ローカルシンボルの表示
ＭＡＰ	マッピングの設定
？ＭＡＰ	マッピングの表示
＿ＭＡＰ	マッピングの解除
ＭＯＶＥ	メモリ内のデータ移動
ＳＲＣＨ	メモリ内のデータサーチ
ＳＵＢＳ	メモリ内容の変更

シンボル制御

ＳＹＭ	シンボルの登録
？ＳＹＭ	シンボルの表示
＿ＳＹＭ	シンボルの削除

ミニアセンブラ

ＡＳＭ	ラインアセンブラ
ＤＡＳＭ	逆アセンブル表示

ロード関係

ＬＯＡＤ	プログラムのロード
ＳＡＶＥ	プログラムのセーブ
ＶＲＦＹ	プログラムのベリファイ

ＩＣＥの実行制御・マンマシンインターフェース

ＡＬＩＡＳ	別名の定義
？ＡＬＩＡＳ	別名の表示
＿ＡＬＩＡＳ	別名の削除
＿ＡＬＩＡＳＵ	ユーザ定義の別名の削除
ＡＵＴＯ	自動再テストの実行
ＣＡＬ	数値計算
ＥＬＯＡＤ	エミュレータ動作環境の回復
ＥＳＡＶＥ	実行環境の退避
ＦＫ	ファンクションキーの設定
？ＦＫ	ファンクションキーの表示
＿ＦＫ	ファンクションキーの解除
ＨＩＳ	コマンド履歴の表示
＿ＨＩＳ	コマンド履歴の削除
ＬＯＧ	ロギング情報の取得開始
？ＬＯＧ	ロギングファイル名の表示
＿ＬＯＧ	ロギング情報の取得終了
ＱＵＩＴ	ＩＣＥの終了
ＲＥＳＥＴ	ＩＣＥの内部初期化
ＳＥＴ	エミュレータ動作環境の設定
？ＳＥＴ	エミュレータ動作環境の表示
ＳＭＰＬ	サンプリングタイマの設定
？ＳＭＰＬ	サンプリングタイマの表示
＿ＳＭＰＬ	サンプリングタイマの削除
：	ＩＣＥコマンドの呼び出しと編集
！	ＩＣＥコマンドの呼び出しと即時実行
＜	コマンドファイルの実行

マクロコマンド制御

CMD	マクロコマンドの登録
？CMD	マクロコマンドの表示
_CMD	マクロコマンドの削除
_CMDU	ユーザ定義のマクロコマンドの削除
ON	ON条件時（ブレーク発生時等）の処理登録
？ON	ON条件時（ブレーク発生時等）の処理表示

ウィンドウ制御

VIEW	テキストファイルの表示
WATCH	ウォッチ変数の登録
_WATCH	ウォッチ変数の削除
INSPECT	インスペクト変数の登録

カバレージ

COV	カバレージ測定条件の設定、表示
？COV	カバレージ測定条件の表示
_COV	カバレージ測定条件の削除

パフォーマンス

PER	パフォーマンス測定条件の設定、表示
？PER	パフォーマンス測定条件の表示
_PER	パフォーマンス測定条件の削除

その他

BELL	ベル音
CAT	ファイルの内容の表示
ECHO	文字列の表示
_ECHO	表示の抑止
REM	コメント行

初期値のコマンドの別名は次の通りです。コマンドの短縮形もコマンドの別名で定義されています。

A	・・・	ASM	D	・・・	DUMP
DA	・・・	DASM	DB	・・・	DUMP
DW	・・・	DUMP／W	E	・・・	SUBS
EB	・・・	SUBS	ED	・・・	SUBS／L
EW	・・・	SUBS／W	EXIT	・・・	QUIT
F	・・・	FILL	G	・・・	GO
KILL	・・・	_SMPL	L	・・・	LOAD
M	・・・	SUBS	PS	・・・	PSTEP
Q	・・・	QUIT	R	・・・	REG
RET	・・・	RETURN	S	・・・	STEP
SR	・・・	SREG	SRC	・・・	DASM／C
SV	・・・	SAVE	U	・・・	DASM／C
V	・・・	VIEW	W	・・・	WATCH
WA	・・・	WATCH／A	WB	・・・	WATCH
WD	・・・	WATCH／L	WL	・・・	WATCH／L
WW	・・・	WATCH／W	Y	・・・	_WATCH
TYPE	・・・	CAT			

　コマンドの別名（短縮形）は、ＩＣＥＩＮＩＴ．ＤＡＴの、ＡＬＩＡＳコマンドで定義されています。ファイルの内容をエディタで書き換えることにより、お客様の好みにあった別名に変更できます。

　コマンドの短縮形を調べたいときは、？ＡＬＩＡＳ（<コマンド名>）のコマンドを入力することにより、別名の表示を行います。

5.9　プログラム実行中に使用できるコマンド

　エミュレーションプログラムが実行中であっても、次のようにメモリアクセス、トレースの設定等を行なうことができます。

(1)メモリアクセス

　オプションのエミュレーションメモリに対して、メモリのリード、ライトコマンド等、次のコマンドを実行できます。

CHNG　DUMP　FILL　MOVE　SRCH　SUBS

(2) バス条件の設定

バスの条件の設定をおこなう次のコマンドを実行できます。

BBn　?BBn　_BBn　EXT　?EXT　_EXT
PASS　?PASS　_PASS
TIM　?TIM　_TIM　TGn　?TGn　_TGn
TRC

(3) パフォーマンス、カバレージの設定

パフォーマンス、カバレージの測定条件の設定を行なう次のコマンドを実行できます。

COV　?COV　_COV　PER　?PER　_PER

(4) その他

?LINE　?SET

６．ブレークおよびトリガ機能

本ＩＣＥは次に述べるような強力なブレークおよびトリガ機能を持っています。
詳細については各コマンドの説明を参照してください。

6.1 ブレーク機能

ソフトブレーク、ＰＣブレーク、バスブレークの３種類のブレーク機能を持っています。

(1) ソフトブレーク
指定した番地を実行する前にブレークします。
本ブレークは指定された番地をＢＲＫ命令（５７７０）に書き換えることにより実現します。従って、ユーザのROM領域には設定できません。
ソフトブレークは１００個まで指定できます。また、パスカウントを使用できます。
ブレークポイントの命令を実行する前にブレークします。

例）
１０００番地を実行する前にブレーク
```
AT 1000
```

２００００番地を１０回（パスカウント）実行する前にブレーク
```
AT 20000,10
```

(2) ＰＣブレーク
指定した番地を実行後ブレークします。
ＲAM、およびＲOM領域に指定できます。
ＰＢ０からＰＢ３の記号で表わし、４本のＰＣブレークを使用できます。

ブレークポイントの命令を実行後にブレークします。

例）
１０００番地を実行したらブレーク
```
PB0 1000
```

１０００番地を１００回実行したらブレーク
```
PB1 1000,100
```

(3) バスブレーク
バス条件によるトリガーによりブレークします。トリガーについては次の項を参照してください。

6.2 トリガ機能

バスブレーク条件（ＢＢコマンド）・・・バスがある条件になったときにブレークします。

トレース条件（ＴＲＣコマンド）・・・トレースの起動、中断および停止条件を指定できます。

タイマ条件（ＴＩMコマンド）・・・タイマ（エミュレーションプログラムの実行時間の計測用）の起動、および停止条件を指定できます。

外部トリガ出力条件（ＥＸＴコマンド）・・・外部トリガの出力条件を指定できます。

パスカウント条件（ＰＡＳＳコマンド）・・・通過回数計測の測定条件を指定できます。

バスブレーク、トレースの取得、エミュレーションプログラムの実行時間計測、外部トリガの出力、通過回数計測を実行することができます。

バスがある条件になったときに、これらの機能を起動、停止することができます。
このバスの条件をトリガ条件といいます。

トリガ条件としては次のものが指定できます。

・プログラムがあるアドレスをプリフェッチした。
・プログラムがあるアドレスをアクセスした。
・プログラムがあるデータをアクセスした。
・ＣＰＵがあるステータスになった。
・あるプローブ信号が入力された。

例として、バスブレークの条件とトリガ条件の関係を説明します。他のトレースの取得、エミュレーションプログラムの実行時間計測、外部信号の出力を実行する条件についても同様のことがいえます。

バスブレークの条件は、単独のトリガ条件、またはトリガ条件の組合せで指定できます。

トリガ条件の組合せとしては次のものが指定できます。

・トリガ条件の成立順序を指定するシーケンシャルブレーク
・トリガ条件の成立回数を指定するパスカウント

トリガ条件は　ＴＧ０　から　ＴＧ３　までの記号で現し４個まで指定できます。
バスブレーク条件はＢＢ０　から　ＢＢ３　までの記号で表し、４個まで指定できます。

例１）

１０００番地をアクセスしたらバスブレークする条件をＢＢ０に、２０００番地を３回アクセスしたらバスブレークする条件をＢＢ１に登録します。

通常このようなバスブレーク条件の設定は次のようにコマンド入力します。
ＢＢ０、ＢＢ１はバスブレーク条件１、２を設定するコマンドです。

(a)

```
＞ＢＢ０　　ＤＡ／１０００
＞ＢＢ１　　ＤＡ／２０００　Ｐ／３
```

ＢＢ０、ＢＢ１はバスブレーク条件１、２を設定するコマンドです。
ＤＡ　はアクセスするアドレスを、Ｐ　はパスカウントを意味します。

これは次のようにそれぞれ２回に分けて指定することもできます。
まず、トリガ条件を設定してから、そのトリガ条件でバスブレーク条件を指定する方法です。

(b)

```
＞ＴＧ０　　ＤＡ／１０００
＞ＢＢ０　　ＴＧ０
＞ＴＧ１　　ＤＡ／２０００
＞ＢＢ１　　３＊ＴＧ１
```

ＴＧ０、ＴＧ１はトリガ条件１、２を設定するコマンドです。

ｍ＊ＴＧｎ　は　トリガ条件　ＴＧｎ　がｍ回成立（パスカウント）することを意味します。

本ＩＣＥは、（ａ）のように指定されても、内部では（ｂ）のようにまずトリガ条件を設定してから、バスブレーク条件の設定をおこないます。
現在設定されているバスブレーク条件の表示を行なうと、次のようにトリガ条件と、バスブレーク条件が表示されます。（ａ）、（ｂ）どちらの形で指定しても同じように表示されます。

```
＞？ＢＢ０
  ＴＧ０　　ＤＡ／１０００
  ＢＢ０　　ＴＧ０

＞？ＢＢ１
  ＴＧ１　　ＤＡ／２０００
  ＢＢ１　　３＊ＴＧ１
```

？ＢＢ０、？ＢＢ１はバスブレーク条件１、２を表示するコマンドです。

例２）
１０００番地を実行してから、２０００番地を３回実行したらブレークするシーケンシャルブレークを設定します。

シーケンシャルブレークの場合、まず、トリガ条件を設定してから、そのトリガ条件でバスブレーク条件を設定します。

```
＞ＴＧ０　　　ＤＡ／１０００
＞ＴＧ１　　　ＤＡ／２０００
＞ＢＢ０　　　ＴＧ０－３＊ＴＧ１
```

－　はシーケンシャルブレークを意味します。

6.3 トリガ条件のキーパラメータ

これらの条件は、次のようなキーパラメータにより指定します。

ＰＡ	プリフェッチアドレス
ＤＡ	プログラムがアクセスするアドレス
Ｄ	プログラムがアクセスするデータ
Ｃ	ＣＰＵのステータス
Ｅ	外部プローブ信号
Ｐ	パスカウント,上記条件の成立回数

(1)　ＰＡ、ＤＡのパラメータ
キーワード　ＰＡ、ＤＡのキーパラメータとして指定できるものは次の通りです。

式	式の値に等しい

(2)　Ｄ、Ｐ、Ｅのパラメータ
キーワード　Ｄ、Ｐ、Ｅ　のキーパラメータとして指定できるものは次の通りです。

式	式の値に等しい

(3) Ｃのパラメータ
キーワード　Ｃ　のキーパラメータとして指定できるものは次の通りです。

Ｒ	データのリード時
Ｗ	データのライト時
ＲＷ	データのリードまたはライト
ＰＦ	プリフェッチ
ＷＯＲＤ	ワードアクセス
ＢＹＴＥ	バイトアクセス
ＤＭＡ	ＤＭＡサイクル
ＲＥＦ	リフレッシュサイクル
ＩＲＯＭ	内蔵ＲＯＭをアクセス
ＩＲＡＭ	内蔵ＲＡＭをアクセス
ＩＲＥＧ	内蔵レジスタをアクセス
ＥＸＴ	外部メモリをアクセス

(4) ドントケアの指定方法

ドントケアは、キーワード　ＰＡ　ＤＡ　Ｄ　に使用することができます。
ドントケアの指定は基数の次にドントケアにしたい所を Ｘ の記号にして表わします。

例）

ＤＡ／Ｈ’１ＸＸＸ　　１６進数の１０００番台をアクセス
ＤＡ／Ｏ’１ＸＸＸ　　８進数の１０００番台をアクセス
ＤＡ／Ｂ’１ＸＸＸ　　２進数の１０００番台をアクセス

注意）　１０進数の基数　Ｄ’はドントケアに使用できません。
ドントケア指定時は３２ビットバス幅のイメージで設定されます。

(5) ＩＣＥが持つハードウウェア資源

バスブレーク条件、トレースブレーク条件、タイマ条件、外部トリガ出力条件において使用できるハードウェア資源は次の通りです。

トリガ条件（ＴＧ0からＴＧ３）	４個
パスカウント	２個
シーケンシャルブレーク	１個（４段）
タイマー	１個

注意）

・既にハードウェア資源が他のトリガ条件により専有されているときは、同一のハードウェア資源を要求するようなトリガ条件は指定できません。
・ブレーク条件が競合（ブレーク条件が同時に発生）したときに、成立したと判断されるブレークの優先順位は次のとおりです。

ソフトブレーク　＞　バスブレーク　＞　ＰＣブレーク

6.4 トレース機能

本ＩＣＥは、次のような強力なトレース機能を用意しています。
コマンドフォーマット等の詳細については、「 ＴＲＣコマンド」を参照してください。

6.4.1 トレースのハード仕様

トレースメモリの大きさは、６４Ｋサイクル分の容量です。
トレースされるデータとしては、アドレスバス、データバス、ＣＰＵステータス、外部プローブ信号があります。

6.4.2 トレーストリガ

トレース取得条件を決定するために、３つのトリガを用意しました。

・ＯＮトリガ　　トレース取得の開始条件を指定します。
・ＯＦＦトリガ　トレース取得の中断条件を指定します。ＯＮトリガが発生すると再びトレースの取得を開始します。
・ＳＴＰトリガ　トレース取得の停止条件を指定します。ＯＮトリガが発生しても、トレースの取得を開始しません。

トリガ条件については「6.2トリガ機能」を参照してください。

6.4.3 トレースモード

以下、コマンドフォーマットの例をあげながら、トレースのモードを説明します。

(1) ＡＬＬモード
ユーザプログラムのＧＯからＢＲＥＡＫまでをトレースします。
トレースバッファフル時に、トレースを継続するか停止するかを指定できます。
トレース停止時に、プログラムをトレースするか継続するかを指定できます。

(a) TRC ALL FUL/CNT

トレースバッファフル時にトレースを継続します。最新の６４Ｋバスサイクルを取得します。
FUL/　は、トレースバッファフル時の制御を指定するキーパラメータです。

```
GO                                                BREAK
|                     |                            |
|                     |                            |
|---------------------|----------------------------|
|                     |                            |
|                     |                            |
                      <---------------------------->
                        トレース取得範囲（６４Kバスサイクル）
```

(b) TRC ALL FUL/BRK END/BRK

トレースバッファフル時にトレースを停止し、プログラムをブレークします。
FUL/　は、トレース取得停止時の制御を指定するキーパラメータです。

```
GO                           BREAK(trace-break)
|     |                            |
|     |                            |
|-----|----------------------------|
|     |                            |
|     |                            |
      <---------------------------->
        トレース取得範囲（６４Kバスサイクル）
```

トレースの最後にＩＣＥのブレーク処理サイクルが含まれます。

(c) TRC ALL FUL/STP END/CNT

トレースバッファフル時にトレースを停止しますが、プログラムはブレークしません。

```
GO                                                        BREAK
|                               |                           |
|                               |                           |
|-------------------------------|---------------------------|
|                               |                           |
|                               |                           |
←──────────────────────────────→
```

トレース取得範囲（６４Kバスサイクル）

(2) ＲＮＧモード

トリガによりトレース取得条件を決めることもできます。また、ディレイサイクルを指定することにより、トレース中断と停止条件をディレイサイクル分遅らすことができます。

(a) TRC RNG ON TG0 OFF TG1 DLY/XX FUL/CNT END/CNT

トレースの取得開始条件（ＯＮトリガ）はＴＧ０のトリガが成立したときで、中断条件（ＯＦＦトリガ）は、ＴＧ１のトリガが成立後ＸＸサイクル経過したときです。（中断条件成立後、ＸＸサイクル内に開始条件が成立しますと、再び、トレースの取得を開始します。）

```
GO                                                        BREAK
| TG0  TG1       TG0    TG1           TG0     TG1          |
|  |    | |       |      | |           |       | |         |
|--|----|-|-------|------|-|-----------|-------|-|---------|
|  |    | |       |      | |           |       | |         |
|        dly              dly                   dly        |
   ←─────→        ←──────→             ←───────→トレース取得範囲
```

(b) TRC RNG ON TG0 FUL/STP END/CNT

トレースの取得開始条件（ＯＮトリガ）はＴＧ０のトリガが成立したときです。
トレースバッファが一杯になるとトレースの取得は行いませんが、プログラムは続行されます。

```
GO                                                   BREAK
|              TG0                                     |
|               |                                      |
|---------------|--------------------------------------|
|               |                                      |
|                                                      |
                ←──────────────────────────────→
```

トレース取得範囲（６４Kバスサイクル）

(c) TRC RNG STP TG0 DLY/XX TG0 FUL/STP

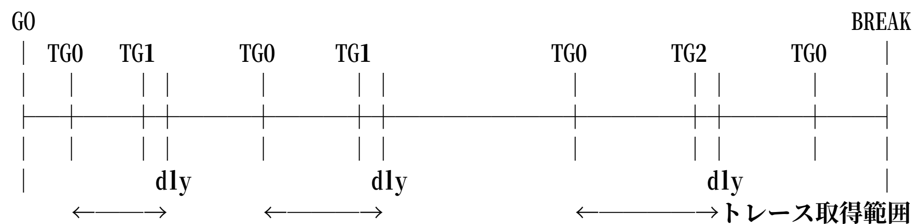
トレースの取得停止条件（ＳＴＯＰトリガ）はＴＧ０のトリガが成立したときです。
条件成立後、ＸＸサイクル後にトレースの取得を停止します。

```
GO                                                   BREAK
|                        TG0                           |
|                         |     |                      |
|-------------------------|-----|----------------------|
|                         |     |                      |
|                              dly                     |
←──────────────────────────────→
```

トレース取得範囲（６４Kバスサイクル）

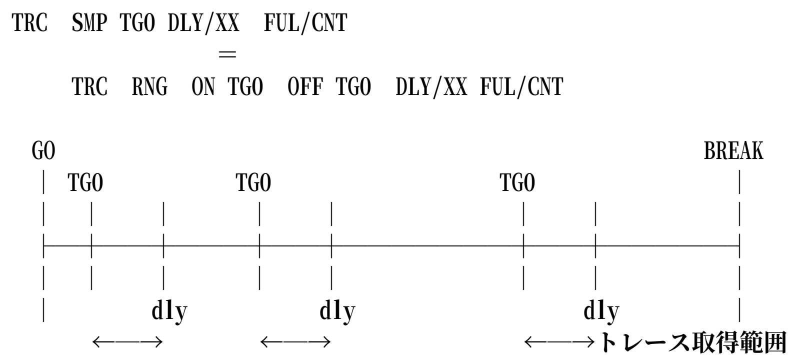
(d) TRC RNG ON TG0 OFF TG1 STP TG2 DLY/XX FUL/STP

トレースの取得開始、中断、および停止のすべての条件を指定したときです。
トレースの停止条件が成立しますと，再びトレースの開始条件が成立しても
トレースの取得は行いません。



(3) ＳＭＰモード

ＳＭＰモードは、あるトリガが成立した瞬間だけトレースするモードです。
ＲＮＧモードの、ＯＮトリガと、ＯＦＦトリガに同一のトリガ条件を設定したのと同じです。



７．パフォーマンスおよびカバレージ機能

本エミュレータには、エミュレーションプログラム内のモジュールの実行時間等を解析するパフォーマンス機能、およびエミュレーションプログラムの実行番地を記録するカバレージ機能があります。
これらの計測はリアルタイムに行われます。

(1) パフォーマンスの機能

同時に、最大８個のモジュール（サブルーチン等）の実行時間、または実行回数を測定することが可能です。
また、あるモジュールの実行時間分布も計測可能です。

(2) カバレージの機能

プログラムが実行した番地を記録する　Ｃ０カバレージ　の測定が可能です。

詳細は、「ＣＯＶコマンド」、および「ＰＥＲコマンド」の説明を参照してください。

８．マッピング機能

ユーザ実機にメモリが無いとき、本ＩＣＥが持つエミュレーションメモリを割り付けることにより、使用することが可能です。

詳細は、「ＭＡＰ　コマンド」の説明を参照してください。

9．各コマンドの詳細

次に各コマンドの詳細を説明します。

書式には、コマンドシンタックス（コマンドの入力形式）が記述されています。
この、書式に使用されている記号の説明を行ないます。

{x｜y｜z}	{} 内のx、y、z　のいずれか１つを指定します。
［x］	[] 内は省略可能です。
x　・・・	パラメータをならべて入力できます。
＜＞	＜＞パラメータを見やすくするためであり特別の意味はありません。

インデックス	コマンド
!、:、＜	!、:、＜
A−C	ALIAS、?ALIAS、_ALIAS、_ALIASU、ASM、AT、?AT、_AT、AUTO、BACK、BELL、BB、?BB、_BB、CAL、CAT、CHNG、CMD、?CMD、_CMD、_CMDU、COME、COV、?COV、_COV
D−F	DASM、DUMP、ECHO、_ECHO、ELOAD、ESAVE、EXT、?EXT、_EXT、FCNV、FILL、FK、?FK、_FK
G−I	GO, GORES, GOW、HIS、_HIS、ICESRC、ICESYM、INSPECT
L−N	LINE、?LINE、LOAD, LOCAL、LOG、?LOG、_LOG、MAP、?MAP、_MAP、MOVE
O−Q	ON、?ON、PASS、?PASS、_PASS、PB、?PB、_PB、PER、?PER、_PER、PSTEP、QUIT
R−S	REG、REM、RESET、RETURN、SAVE、SET、?SET、SMPL、?SMPL、_SMPL、SRCH、SREG、STEP、STOP、SUBS、SYM、?SYM、_SYM
T−W	TDMP、TG、?TG、_TG、TIM、?TIM、_TIM、TRC、?TRC、_TRC、VIEW、VRFY、WATCH、_WATCH

！　　　　　ヒストリ内にあるコマンドの実行

書式　！　［　文字列　］

機能　コマンドヒストリに記憶されているコマンドの実行を行います。

解説　コマンドヒストリ（コマンドの履歴）に記憶されているコマンドを次のように捜しだして実行します。

！文字列	最後に実行された文字列で始まるコマンドを実行します。
！	直前に入力したコマンドを実行します。

ＨＩＳコマンドによりコマンドヒストリを表示することができます。

例

最後に実行したＰＢＯコマンドを再び実行します。

```
>!PBO
```

書式　：　［　文字列　］

機能　コマンドヒストリに記憶されているコマンドの編集と実行を行います。

解説　コマンドヒストリ（コマンドの履歴）に記憶されているコマンドを次のように捜しだして編集後、キャリッジリターン入力により実行します。

：文字列　　　最後に実行された文字列で始まるコマンドを表示し、
編集とコマンド入力が可能となります。

：　　　　　直前に入力したコマンドを表示し、編集とコマンド
入力が可能となります。

ＨＩＳコマンドによりコマンドヒストリを表示することができます。

例

最後に実行したＰＢ１コマンドを再び編集後、実行します。
>:PB1

<　　　　　　コマンドファイルの実行

書式　＜　＜ファイル名＞

機能　コマンドファイルの実行を行います。

解説　指定された＜ファイル名＞をコマンドファイルとして実行します。

　コマンドファイルは、ＩＣＥのコマンドをキー入力するときと同じようにコマンドやパラメータを並べたものです。

注意）
・次のコマンドは、コマンドファイルでは実行できません。
　＜
　ＡＵＴＯ
　＿ＡＬＩＡＳＵ
　＿ＣＭＤＵ

・ＧＯ、ＧＯＲＥＳコマンドをコマンドファイルで使用するときは、それぞれ各コマンドの説明を参照してください。

・ＬＯＧコマンド　で取得したロギングファイルを、コマンドファイルとして実行することも可能です。

例

レジスタＲ０、Ｒ１にある値を設定してからのエミュレーションプログラムの実行を、コマンドファイルで行います。

```
>CAT CFILE                              (コマンドファイルの内容を表示します。)
SREG  R0/1234 R2/5678
GO
> < CFILE                               (コマンドファイルを実行します。)
```

ロギングファイルにロギングを取得しますと　＜　コマンドでも　ＡＵＴＯコマンドでも実行可能です。

```
>LOG ALOG                               (ロギングの取得を開始します。)
>AT %SUBX %SUBY
>GO
>_LOG                                   (ロギングの取得を終了します。)
............
>< ALOG.LOG                             (ロギングファイルのコマンド入力の記録ファイルを、
                                         コマンドファイルとして実行します。)
```

書式　ＡＬＩＡＳ　［<別名>　<文字列>］

機能　指定したコマンドの文字列に対して別名の定義を行います。

解説　指定した<文字列>に対して<別名>の定義を行います。
これ以降、コマンド入力で別名の文字列を入力しますと指定された文字列に置き換えて実行します。

パラメータを省略しますと、ＡＬＩＡＳ>のプロンプトを表示しますので続いて<別名>　<文字列>　と入力することにより別名の定義を行うことができます。
例を参照してください。

例

１０００番地から１ＦＦＦ番地まで０クリアするコマンドをＣＬＥＡＲの文字列に定義します。

```
>ALIAS CLEAR FILL 1000 1FFF 0
```

上記コマンド入力後　ＣＬＥＡＲ　と入力しますと　ＦＩＬＬ　１０００　１ＦＦＦ　０　の文字列に置き換えて実行しますので、１０００番地から１ＦＦＦ番地まで０クリアされます。

１０００番地から実行するコマンドを　ＲＵＮ　の文字列に置き換えます。

```
>ALIAS RUN GO 1000
```

連続して別名の定義を行います。

```
>ALIAS
ALIAS> P0 PB0
ALIAS> P1 PB1
....
ALIAS>                  （キャリッジリターンの入力でコマンドを終了します。）
>
```

書式　？ＡＬＩＡＳ　［　＜別名＞　・・・　］

機能　別名に対して定義ずみの文字列を表示します。

解説　別名に対して定義ずみの文字列を表示します。
別名を省略しますと、すべての定義ずみの別名を表示します。

別名が未定義のときは、ａｌｉａｓ　ｎｏｔ　ｆｏｕｎｄ　が表示されます。

例

ＣＬＥＡＲ　と　ＲＵＮ　の別名コマンドに定義されている文字列を表示します。

```
>?ALIAS CLEAR RUN
```

書式　＿＿ＡＬＩＡＳ　［　＜別名＞　・・・　］

機能　別名の定義を解除します。

解説　別名の定義を解除します。
別名を省略しますと、すべての定義ずみの文字列を解除します。

注意）
・別名を省略しますと本ＩＣＥが立ち上げ時に（ＩＣＥＩＮＩＴ．ＤＡＴで）定義した別名も解除されます。
ユーザが定義したもの全てを解除したいときは、＿＿ＡＬＩＡＳＵコマンドを使用してください。

例

定義済みのＣＬＥＡＲ　と　ＲＵＮ　の別名コマンドを解除します。

```
>_ALIAS CLEAR RUN
```

書式

＿ＡＬＩＡＳＵ

機能

ユーザが設定した別名の定義を全て解除します。

解説

ユーザが設定した別名の定義を全て解除します。

注意）
　本コマンドは　ＡＬＩＡＳコマンドにより定義されています。

例

ユーザが設定した別名の定義を全て解除します。

```
>_ALIASU
```

書式　ＡＳＭ ［ ＜ アドレス ＞ ］

機能　ニーモニックのアセンブルを行います。

解説　ＡＳMコマンドはニーモニックをアセンブルして、その結果である機械語を、指定した<アドレス>に書き込みます。

コマンド実行中はアドレスを表示しますので、続けてニーモニックを入力できます。

コマンドを終了したいときは、 ．＜ＣＲ＞を入力します。

オペランドにシンボルを使用するときは、シンボル名の前に %を付けてください。

オペランドには、加算 ＋ と 減算 － の演算が行えます。

ニーモニック入力後、コードウィンドウに入力したアセンブル命令と機械語に修正して表示されます。

注意）

(1) ロケーションカウンタの参照はできません。

(2) 「－」記号を単項演算子としてオペランドに使用することはできません。

(3) 制御命令は使用できません。

(4) 命令オペランドで指定するデータの基数は、１６進数です。

例

１０００番地からアセンブル入力していきます。

```
>A 1000
001000 MOV @1100,R1
001004 MOV R2,R3
001006 JSR @R1
001008 .
>
```

シンボルを使用してアセンブル入力していきます。

```
>SYM SYM1 2100
>SYM SYM2 2000
A 2000
002000 MOV @%SYM1+2,R4
002004 MOV @%SYM2+D'10,R5
002008 .
>
```

書式　ＡＴ　　＜アドレス＞［、＜回数＞］　　・・・

機能　ソフトブレークポイントの設定を行います。

解説　指定したアドレスにソフトブレークポイントを作成します。
プログラム実行時ブレークポイントに達すると、ブレークポイントの命令を実行する前に実行を停止します。

ブレークポイントは、最大１００個まで登録できます。

＜回数＞は、ブレークポイントが成立するまでの回数を指定します。
＜回数＞を指定しないと、回数＝１が採られます。成立回数は、プログラムでカウントします。
＜回数＞は最大６５５３５回です。
（基数を省略したとき１０進数がとられます。）

実行停止後、ＯＮ　ＡＴ　コマンドで定義された処理（ＯＮＡＴマクロコマンド）が実行されます。ＯＮ　ＡＴ　コマンドの初期値は実行後のレジスタの値と次に実行する命令を表示するようになっています。

ソフトブレークポイントは、内蔵レジスタ空間、ユーザのＲＯＭ空間以外の空間に設定できます。エミュレーシュンＲＯＭ空間や内蔵ＲＯＭ空間にも使用できます。（「ＭＡＰコマンド」を参照してください。）

例

関数名ｔｔｓｕｂに　ソフトブレークを設定します。
```
>AT %ttsub
```

現在のＰＣ＋２番地を６回実行するとソフトブレークします。
```
>AT  .PC+2,6
```

書式　？ＡＴ

機能　ソフトブレークポイントの表示を行います。

解説　全ソフトブレークポイントの条件の表示を行います。

　一度ソフトブレーク条件をファイルに退避して、再び、ファイルに退避したブレーク条件を設定することが可能です。例を参照してください。

例

全てのソフトブレークポイントを表示します。

```
>?AT
AT H'002000,1 H'001000,20 H'010000,1 H'020000,1   ...
AT H'003000,1 H'004000,1
```

一度ソフトブレーク条件をファイルに退避して、再び、ファイルに退避したブレーク条件を設定します。

```
>?AT > BRKAT.SAV         （ソフトブレーク条件をファイルにセーブします。）
................         （いろいろなコマンドを実行します。）
>< BRKAT.SAV             （ファイルにセーブしたソフトブレーク条件を設定します。）
```

書式　＿ＡＴ　　　［＜アドレス＞・・・］

機能　ソフトブレークポイントの削除を行います。

解説　指定したソフトブレークポイントの条件の削除を行います。

　＜アドレス＞を指定しますと、そのアドレスに設定されているソフトブレークポイントを削除します。
　＜アドレス＞を指定しませんと、全てのソフトブレークポイントを削除します。

例

１０００番地と２０００番地にソフトブレークポイントを設定しその後削除します。

```
>AT 1000 2000,10
  ..........
>_AT 1000 2000
```

書式　ＡＵＴＯ　＜ファイル名＞

機能　自動再テストを実行します。

解説　一度行ったテストを自動的に再テストを行います。

〈ファイル名〉には、ＬＯＧコマンドで指定したロギングファイル名を指定します。
例を参照してください。

本コマンドは、指定したロギングファイル名に拡張子　．ＬＯＧを付加したファイルをオープンします。そのファイルの内容を読み込み、自動的にＩＣＥのコマンドとして実行します。

注意）
・次のコマンドを含むものは、ＡＵＴＯコマンドでは実行できません。
　＜
　ＡＵＴＯ
　＿ＡＬＩＡＳＵ
　＿ＣＭＤＵ

例

一度実行したコマンドを　ロギングファイル　に記録し再び自動的に再実行します。

```
>LOG LOGSAV              (ロギングを取得します。ファイル名に拡張子は付けません。)
..........               (いろいろなＩＣＥのコマンドを実行します)
>_LOG                    (ロギングの取得を終了します。)
>AUTO LOGSAV             (自動再テストを実行します。)
```

書式　ＢＡＣＫ

機能　関数の戻り番地を表示します。

解説　Ｃ言語で記述した関数の戻り番地をトレースしてリターンする順番にコマンドウィンドウに表示します。

表示形式：　<関数名>＋ＸＸ（引数）

＜関数名＞	呼出し元の関数名
ＸＸ	＜関数名＞からの相対アドレス
引数	ＰＣが示している関数の引数の値

オプションウィンドウのバックトレース表示では、本コマンドと同じ内容をオプションウィンドウのサイズ内で表示します。

戻りアドレスは、一般にスタックエリアに格納されているため、関数の初めと終わりの部分では、正しい戻りアドレスが表示されない場合があります。

例

関数 func_sub() まで実行して、関数呼出し順を確認する場合。

```
>AT %func_sub
>GO
>BACK
```

書式　ＢＥＬＬ

機能　ベル音を発生します。

解説　ベル音を発生します。

例

ベル音を発生します。

```
>BELL
```

書式 (1) ＢＢｎ
［ＰＡ／<プログラムアドレス>］
［ＤＡ／<データアドレス>］
［Ｄ／<データ>］
［Ｃ／<ＣＰＵステータス>］
［Ｅ／<外部プローブ>］
［Ｐ／<パスカウント>］

(2) ＢＢｎ
［<パスカウント>＊］ＴＧｍ［－［<パスカウント>＊］ＴＧｍ　・・・］
（　ＴＧｍの論理式です　）

ｎ　　：ブレークポイントの番号（０から３）
ｍ　　：トリガの番号（０から３）

機能 バスブレークポイントの設定を行います。

解説 指定した条件をバスブレーク条件として登録します。

ブレークポイントは、最大４個まで登録できます（０から３）。
実行停止後、ＯＮ　ＢＢｎ　コマンドで定義された処理（ＯＮＢＢｎマクロコマンド）が実行されます。初期値は実行後のレジスタの値と次に実行する命令を表示するようになっています。

(1) 書式(1)に関して

各キーワードの意味は、次の通りです。

ＰＡ　　プログラムの実行アドレス（プリフェチ時）

ＤＡ　　プログラムがアクセスするアドレス

Ｄ　　プログラムがアクセスするデータ

Ｃ　　ＣＰＵのステータス
ＣＰＵステータスの設定はＴＧ（トリガ条件設定）
コマンドを参照して下さい。

Ｅ　　外部信号プローブの値
１バイトの値です。

Ｐ　　パスカウント,上記条件の成立回数
パスカウントは、最大６５５３６回です。
（基数を省略したとき１０進数がとられます。）

キーワード　ＰＡ、ＤＡ、Ｄには、ドントケアの指定を行うことができます。

例）
ＤＡ／Ｈ’１ＸＸＸ　　１６進数の１０００番台をアクセス
したらブレーク

(2) 書式(2)に関して

ＴＧｍは、ＴＧコマンドで定義したトリガ条件を指定します。
ＴＧｍの論理式として次が使用できます。

(a) シーケンシャルブレーク
－　で連結することによりシーケンシャルブレークを４段まで
設定できます。

(b) パスカウント

<パスカウント>＊ＴGm　の形式により、トリガ条件ＴGm
の成立回数（最大６５５３６回）を設定できます。
なお、パスカウントは、上記(1) の書式の　Ｐ／　キーパラメータでも指定可能です。

例）

トリガ条件　ＴＧ０　が３回実行後　ＴＧ１　を実行したときにブレーク成立を設定します。

```
>BB0  3*TG0-TG1
```

バスブレーク発生後、ＯＮ　ＢＢnコマンドで定義された処理（ＯＮＢＢn　マクロコマンド）が実行されます。初期値は実行後のレジスタの値と次に実行する命令を表示するようになっています。

注意）

・シーケンシャルブレークの最後に成立するトリガ条件は、他のブレークで最後に成立するトリガ条件でないものを指定してください。
・本コマンドを使用しますと２本の（ハードウエア）パスカウンタの現在の計測値は（設定時にエラーが発生しても）０にクリヤされます。

例

１０００番地に５０をライトしたときにブレークします。

```
>BB0 DA/1000 D/50 C/W
```

外部入力信号が１６進数の　ＣＦ　の値になったときにブレークします。

```
>BB2 E/CF
```

一度バスブレーク条件をファイルに退避して、再び、ファイルに退避したブレーク条件を設定します。

```
>?BB > BRKBB.SAV              (バスブレーク条件をファイルにセーブします。)
........................      (いろいろなコマンドを実行します。)件を設定します。)
><BRKBB.SAV                   (ファイルにセーブしたブレーク条
```

書式　？ＢＢ［ｎ］

機能　バスブレークポイントの表示を行います。

解説　指定したバスブレークポイントの条件の表示を行います。

ｎ（ブレークポイント番号）を指定しないと全てのバスブレーク条件を、表示します。

表示は、まずトリガ条件が表示され、続いてバスブレーク条件とトリガ条件の関係を論理式で表示します。

例

全てのバスブレーク条件を表示します。

```
>?BB
TG0 DA=2000 C=R
BB0 TG0
TG1 D=FF C=W
TG2 DA=4000
BB1 3*TG1-TG2
```

一度バスブレーク条件をファイルに退避して、再び、ファイルに退避したブレーク条件を設定します。

```
>?BB > BRKBB.SAV            （バスブレーク条件をファイルにセーブ）
.........................   （いろいろなコマンドを実行）
><BRKBB.SAV                 （ファイルにセーブしたブレーク条件を設定します。）
```

書式　**＿ＢＢ［ｎ］**

機能　**バスブレークポイントの削除を行います。**

解説　**指定したバスブレークポイントの条件の削除を行います。**

ｎ（ブレークポイント番号）を指定しないと全てのバスブレーク条件を削除します。

例

１番目のバスブレークポイントを削除します。

```
>_BB1
```

書式　ＢＥＬＬ

機能　ベル音を発生します。

解説　ベル音を発生します。

例

ベル音を発生します。
```
>BELL
```

書式

(1) ＣＡＬ　　　＜　式　＞
(2) ＣＡＬ／Ｅ　{＃｜%}
　　%　　式のシンボルの前に%の付加が必要です。
　　＃　　式のシンボルの前に%の付加が不要です。

機能

式の計算を行い、結果を各表現で表示します。

解説

（１）　式の計算を行い、結果を１０進数、１６進数、２進数、およびアスキーコードで表示します。
　式として使用できるものについては、「5．2　式」を参照してください。結果は全て４バイトのデータとして扱います。

（２）　コマンドパラメータでの式の評価において、シンボルの指定に%の付加が必要か、不要かを指定します。例を参照してください。

ＣＡＬ／Ｅ　%　　　シンボルの前に%の付加が必要です。

ＣＡＬ／Ｅ　＃　　　シンボルの前に%の付加が不要です。英数字で始まる１６進数の指定には０を先頭に付加します。
このモードでは乗算（＊）は使用できません。

例

Ｒ０レジスタとＲ１レジスタの内容を加え結果を表示します。

```
>CAL .R0+.R1
DECIMAL        : 65535
HEXA           : 0000FFFF
BINARY         : 00000000000000001111111111111111
ASCII          : '....'
```

シンボル指定での%の付加の必要、不要を制御します。

```
>cal/e %        シンボルの前に%が必要です。（初期値）
>dasm  %main    関数ｍａｉｎを逆アセンブルします。
>dump  abc      ＡＢＣ（１６進数）番地からダンプします。

>cal/e #        シンボルの前に%が不要です。
>dasm  main     関数ｍａｉｎを逆アセンブルします。
>dasm  %main    %が付加されていてもシンボルとして扱います。
>dump  abc      シンボルａｂｃからダンプします。
                （シンボル未登録ならエラー）
>dump  0abc     ＡＢＣ（１６進数）番地からを指定するときは先頭に０を付加してください。
>subs  sub_1*2  乗算（＊）はエラーとなりますので%の付加が必要なモードで実行してください。
```

書式　ＣＡＴ　　　＜　ファイル名　＞

機能　ファイルの内容を表示します。

解説　指定したファイルの内容を表示します。

例

ファイル名　ＦＩＬＥ．ＤＡＴ　の内容を表示します。

```
>CAT FILE.DAT
```

書式　ＣＨＮＧ［｛／Ｂ｜／Ｗ｜／Ｌ｜／？｝］＜レンジ＞＜旧データ＞＜新データ＞

Ｂ　：　１バイト単位に変更（省略値）
Ｗ　：　２バイト単位に変更
Ｌ　：　４バイト単位に変更
？　：　変更の確認付き

機能　メモリのある範囲のデータをサーチし一致するものを変更します。

解説　指定した＜レンジ＞の範囲のデータ内で、＜旧データ＞と一致するものを捜し＜新データ＞に変更します。

オプションの　Ｂ、Ｗ、Ｌにより　バイト、ワード、ロングワード単位での変更を指定します。
省略値は　Ｂ（バイト）となります。

？　オプションにより次のようなメッセージが表示されますので、変更の確認をとりながら、メモリ内容の変更を行うことができます。

番地　：　旧データ値　－＞　新データ値　（Ｙ／Ｎ／Ｑ）　？＞

Ｙを入力　・・・　変更を実行します。
Ｎを入力　・・・　変更は行わず次のデータをサーチします。
Ｑを入力　・・・　変更は行わず処理を終了します。

例

１０００番地から２ＦＦＦ番地までの　ＡＡ　の値を　ＢＢ　に変更します。

```
>CHNG 1000 2FFF AA BB
```

１０００番地から２ＦＦＦ番地までの　ＡＡ　の値を　ＢＢ　に確認をとりながら変更します。

```
>CHNG/? 1000 2FFF AA BB
001010:AA->BB(Y/N/Q) ? > Y              (変更します。)
001020:AA->BB(Y/N/Q) ? > N              (変更しません。)
001030:AA->BB(Y/N/Q) ? > Y              (変更します。)
001040:AA->BB(Y/N/Q) ? > Q              (処理を終了します。)
>
```

書式　ＣＭＤ　＜マクロコマンド名＞

機能　マクロコマンドを登録します。

解説　指定したコマンド名をマクロコマンドとして登録します。

　指定した〈マクロコマンド名〉が既に登録されているときは、コマンドを再定義し、先に指定したコマンドを無効とします。

　本コマンドを入力しますとエミュレータはプロンプトとして　？　を表示します。続いてコマンドを入力できます。
　＜ＣＲ＞を入力しますと次のプロンプトが表示されます。

　．の入力で１つのマクロコマンドの登録を終了します。
　続いてｍａｃｒｏ－ｎａｍｅ？の表示がなされますので、続けて、マクロコマンドを登録するときは、次に登録したいマクロコマンド名を入力します。ＣＭＤコマンドを終了したい時はもう一度　．を入力します。

注意）
・マクロコマンド内からマクロコマンドを実行することはできません。
・次のコマンドはマクロコマンド内で使用できません。
　　＜　、　＿ＡＬＩＡＳＵ　、ＣＭＤ　、＿ＣＭＤＵ
・別名で定義されているコマンド名を、マクロコマンド内で使用することはできません。
　元のコマンド名を指定してください。

例

DUMP,REG,STEPコマンドを実行します。

```
>CMD DRS
?DUMP
?REG
?STEP
?.
?macro_name?.
>
```

書式　？CMD　［　＜マクロコマンド名＞　・・・　］

機能　マクロコマンドの定義内容を表示します。

解説　指定したマクロコマンド名の定義体を表示します。

＜マクロコマンド名＞を指定しませんと全てのマクロコマンドについて表示します。

入力先のリダイレクト機能により、ファイルにセーブしたマクロコマンドを、再びマクロコマンドとして再設定することも可能です。

例

マクロコマンドを表示します。

```
>?CMD   sd
sd
        reg
        dump
.                         （一つのマクロコマンドの表示の終了を意味します。）
.                         （マクロコマンドの表示の終了を意味します。）
```

書式　＿＿ＣＭＤ　［　＜マクロコマンド名＞　・・・　］

機能　マクロコマンドの定義を削除します。

解説　指定したマクロコマンド名の定義を削除します。

＜マクロコマンド名＞を指定しませんと全てのマクロコマンドについて削除します。

注意）
・マクロコマンド名を省略しますと本ＩＣＥが立ち上げ時に（ＩＣＥＩＮＩＴ．ＤＡＴで）
　定義したマクロコマンドも削除されます。ユーザが定義したもの全てを解除したいときは、
　＿＿ＣＭＤＵコマンドを使用してください。

例

定義済みのＣＬＥＡＲ　のマクロコマンドを解除します。

```
>_CMD CLEAR
```

書式　**＿＿ＣＭＤＵ**

機能　**ユーザが設定したマクロコマンドの定義を全て解除します。**

解説　**ユーザが設定したマクロコマンドの定義を全て解除します。**

注意）
　本コマンドは　ＡＬＩＡＳコマンドにより定義されています。

例

ユーザが設定したマクロコマンドの定義を全て解除します。

```
>_CMDU
```

書式　COME　[< ブレークポイントアドレス >]

機能　ブレークポイントアドレスの前までプログラムを実行します。

解説　指定した＜ブレークポイントアドレス＞を設定してからから、ユーザプログラムを実行し、＜ブレークポイントアドレス＞を実行する前にブレークします。

注意）
　ブレークポイントは、ソフトブレークを使用します。従ってユーザ実機のＲＯＭ領域内には＜ブレークポイントアドレス＞は設定できません。

例

１２３４番地の前まで実行します。

```
>COME  1234
```

書式
(1) COV　<レンジ>
(2) COV　［ON｜OFF｜CLR］
　ON　：　測定開始
　OFF　：　測定停止
　CLR　：　測定結果クリア
(3) COV　DMP　［<レンジ>］
(4) COV　{AND｜OR}　<レンジ>
(5) COV　{SAVE｜LOAD}　<ファイル名>
　SAVE：　測定結果セーブ
　LOAD：　測定結果ロード

機能
カバレージの測定条件（範囲）の設定や、表示を行います。

解説
カバレージ測定条件（範囲）の設定、測定結果の表示、および測定結果をファイルにセーブ、またはファイルからロードします。

コマンドのパラメータにより次の様な機能を持ちます。

（1）COV　<レンジ>

カバレージの測定範囲を設定します。
カバレージの測定範囲は１２８Kバイトの境界で設定されます。
カバレージの測定範囲は１２８Kバイト単位で２ブロックまで設定できます。
<最終番地>を省略しますと、<先頭番地>を含む１２８Kバイトの境界の範囲に設定されます。

例）
COV　8000
・・・　０番地から１FFFF番地の１２８Kバイトの範囲をカバレージ測定します。

COV　2000　31FFF
・・・　０番地から３FFFF番地の２５６Kバイトの範囲をカバレージ測定します。

（2）COV　［ON｜OFF｜CLR］

カバレージ測定の開始、停止、測定結果のクリアを指定します。

ON　・・・　カバレージの測定を開始します。
OFF・・・　カバレージの測定を停止します。
CLR・・・　カバレージの測定結果をクリアします。

（3）COV　DMP　［<レンジ>］

カバレージの測定結果をダンプ形式で表示します。

（4）　COV　{AND｜OR}　<レンジ>

指定した　<レンジ>　の範囲をプログラムが実行したかどうかを表示します。

AND　・・・　<レンジ>の範囲を全て実行していると次のように表示されます。

and　［<先頭番地>－<最終番地>］　HIT

<レンジ>の範囲の一部でも実行されていなければ、なにも表示されません。

ＯＲ　　・・・　＜レンジ＞の範囲の一部でも実行していると次のように表示されます。

ｏｒ　［＜先頭番地＞－＜最終番地＞］　ＨＩＴ

＜レンジ＞の範囲が全く実行されていなければなにも表示されません。

（５）　ＣＯＶ　｛ＬＯＡＤ｜ＳＡＶＥ｝　＜ファイル名＞

カバレージの測定結果を、指定したファイルから読み込んだり（ロード）、書き込んだり（セーブ）します。

ＳＡＶＥ　・・・　カバレージの測定結果をファイルにセーブします。
カバレージの測定範囲もセーブされます。

ＬＯＡＤ　・・・　カバレージの測定結果をファイルからロードします。
測定範囲もセーブ時の条件に再設定されます。

プログラム変更後アドレスが変わった場合セーブしたデータを使用しないでください。

例

カバレージの測定を行い、一度ＩＣＥを終了してから、後日再び再測定を行います。

```
>COV %SUB_START %SUB1_END-1        (カバレージの測定範囲を設定します。)
>COV ON                            (カバレージの測定を開始します。)
>GO                                (ユーザプログラムを実行します。
                                    実行したアドレスがカバレージに記録されます。)
>COV OFF                           (カバレージの測定を停止します。)
>GO                                (ユーザプログラムを実行します。
                                    実行したアドレスはカバレージに記録されません。)
>COV DMP %SUB_START %SUB1_END-1    (カバレージの測定結果をダンプ形式で表示します。)
>COV SAVE COVERAGE.SAV             (カバレージの測定結果をファイルにセーブします。)
>Q                                 (一度ＩＣＥを終了します。)
(再びＩＣＥを起動します。)
>COV %SUB_START %SUB1_END-1        (カバレージの測定範囲を設定します。)
>COV LOAD COVERAGE.SAV             (先に、セーブしたカバレージの測定結果と測定範囲
                                    をロードします。)
>COV ON                            (カバレージの測定を開始します。)
>GO                                (ユーザプログラムを実行します。
                                    実行したアドレスはカバレージに追加記録されます。)
```

書式　？ＣＯＶ

機能　カバレージの測定条件（範囲）を表示します。

解説　カバレージの測定条件（範囲）を表示します。

例

カバレージの測定範囲を表示します。

```
>?COV
COV  H'10000 H'10FFF
```

書式　　**＿ＣＯＶ**

機能　　**カバレージの測定条件を解除します。**

解説　　**カバレージの測定条件を解除します。**

例

カバレージの測定条件を解除します。

```
>_COV
```

書式　(1) ＤＡＳＭ［　＜　レンジ　＞｜＜アドレス＞　］
(2) ＤＡＳＭ／Ｃ　［＜アドレス＞　］

機能　メモリ内容を逆アセンブル表示します。

解説　指定されたメモリ内容を逆アセンブルして表示します。

書式により次を実行します。

(1) ＤＡＳＭ　［　＜　レンジ　＞｜＜アドレス＞　］
指定された＜レンジ＞を、逆アセンブル表示します。
＜アドレス＞を指定しますと、＜アドレス＞から８行分の逆アセンブル表示を行います。
パラメータを指定しませんと次のメモリ内容の逆アセンブル表示を８行分行います。

(2) ＤＡＳＭ／Ｃ　［＜アドレス＞　］
オプション　Ｃ　を指定しますと混在モードウィンドウを指定した＜アドレス＞より書き換えます。

分岐先のアドレスは、シンボル（最大２０文字まで）に変換されて表示されます。
（例 BSR 1C0 symbol:24）

補注）
ＤＡＳＭ／Ｃは、ＳＲＣ，Ｕの別名に定義されています。
ＤＡＳＭはＤＡに定義されています。

例

シンボル　main　から逆アセンブルします.

```
>da %main
```

現在のプログラムカウンタより８行分逆アセンブルします.

```
>DA .PC
```

サブルーチン　ＸＸＸ＿ＳＵＢ　をコードウィンドウに表示します。

```
>SRC %XXX_SUB
```

書式　DUMP［{／B｜／W｜／L｜／D}］　［{<レンジ >｜<アドレス>}］

B ： １バイト単位で表示（バイトアクセス）
W ： ２バイト単位で表示（ワードアクセス）
L ： ４バイト単位で表示（ロングアクセス）
D ： １０進数で表示

機能　メモリ内容のダンプを行います。

解説　指定した、<レンジ>分のメモリ内容の表示を行います。

　<アドレス>を指定しますと、<アドレス>から１２８バイト分のメモリ内容のダンプを行います。

　パラメータを省略しますと、先のダンプコマンドで表示した次の１２８バイト分のメモリ内容のダンプを行います。

　オプションにより次の機能を実行します。

B ・・・ バイト（１バイト）単位でリードし表示します。（省略値）
W ・・・ ワード（２バイト）単位でリードし表示します。
L ・・・ ロング（４バイト）単位でリードし表示します。

例

シンボル　ｓｔａｒｔ　番地からダンプします。
```
>d %start
```

次のようにサイズを指定しても入力できます。
```
>D 1000 \7F
```

次の１２８バイトを表示します。
```
>D
```

書式　ＥＣＨＯ　［＜文字列＞　・・・］

機能　文字列をコマンドウィンドウに表示します。

解説　指定した＜文字列＞をコマンドウィンドウに表示します。

空白は、”記号で囲むことにより使用することができます。
改行は　￥ｎ　の記号により行われます。

パラメータの文字列を指定しませんと　＿ＥＣＨＯ　コマンドで指定した　表示の抑止を解除します。

例

文字列を表示します。

```
>ECHO "<THIS IS THE MESSAGE>\n"
```

コマンドウィンドウへの表示を抑止し再び解除します。

```
>_ECHO
                    （コマンド入力しても、何も表示されません）
( ECHO )            （ＥＣＨＯコマンドを入力して表示の抑止を解除します。
                      ＥＣＨＯコマンド自体も表示されません。）
>                   （通常の、エコーバックや表示がされるモードになります。）
```

書式　＿ＥＣＨＯ

機能　コマンドウィンドウへの表示を抑止します。

解説　コマンドウィンドウでのエコーバックやコマンドの実行結果の表示を抑止します。

ＥＣＨＯコマンドで解除することができます。

例

コマンドウィンドウへの表示を抑止し再び解除します。

```
>_ECHO
                    （コマンド入力しても、何も表示されません）
( ECHO )            （ＥＣＨＯコマンドを入力して表示の抑止を解除します。
                      ＥＣＨＯコマンド自体も表示されません。）
>                   （通常の、エコーバックや表示がされるモードになります。）
```

書式　ＥＬＯＡＤ

機能　エミュレータの動作環境を回復します。

解説　ＥＳＡＶＥでファイルに退避したエミュレータの動作環境を回復します。

回復するエミュレータの動作環境は次のとおりです。

外部信号制御	（ＬＩＮＥコマンド）
エミュレータ動作環境	（ＳＥＴコマンド）
ソフトブレーク	（ＡＴコマンド）
ＰＣブレーク	（ＰＢコマンド）
バスブレーク	（ＢＢコマンド）
マッピング	（ＭＡＰコマンド）
トレース	（ＴＲＣコマンド）
外部トリガ出力	（ＥＸＴコマンド）
実行時間計測	（ＴＩＭコマンド）

ＥＳＡＶＥコマンド実行後終了し、その後エミュレータプログラムを起動し、ＥＬＯＡＤコマンドを入力しますと、先のエミュレータを終了したときの動作環境が回復できます。

補注）
本コマンドを実行すると上記の条件を　ＩＣＥＳＡＶＥ．＄＄＄よりリードします。

例

ＥＳＡＶＥコマンドで動作環境を退避して終了します。再び、エミュレータを起動後、ＥＬＯＡＤコマンドで同一環境に再設定します。

```
>ESAVE                    （動作環境を退避して終了します。）
>Q
```

（再びエミュレータを起動します。）

```
>ELOAD                    （先にエミュレータを使用していたときと同一環境にします。）
```

書式　ＥＳＡＶＥ

機能　現在のエミュレータの実行環境を退避します。

解説　現在のエミュレータの実行環境をファイルに退避します。

退避するエミュレータの動作環境は次のとおりです。

外部信号制御	（ＬＩＮＥコマンド）
エミュレータ動作環境	（ＳＥＴコマンド）
ソフトブレーク	（ＡＴコマンド）
ＰＣブレーク	（ＰＢコマンド）
バスブレーク	（ＢＢコマンド）
マッピング	（ＭＡＰコマンド）
トレース	（ＴＲＣコマンド）
外部トリガ出力	（ＥＸＴコマンド）
実行時間計測	（ＴＩＭコマンド）

ＥＳＡＶＥコマンド実行後終了し、その後エミュレータプログラムを起動し、ＥＬＯＡＤコマンドを入力しますと、先のエミュレータを終了したときの動作環境が回復できます。

補注）
・本コマンドを実行すると上記の条件を　ＩＣＥＳＡＶＥ．＄＄＄にセーブします。
・本コマンドは、マクロコマンドです。

例

ＥＳＡＶＥコマンドで動作環境を退避して終了します。再び、エミュレータを起動後、ＥＬＯＡＤコマンドで同一環境に再設定します。

```
>ESAVE                    (動作環境を退避して終了します。)
>Q
```

（再びエミュレータを起動します。）

```
>ELOAD                    (先にエミュレータを使用していたときと同一環境にします。)
```

書式　(1) ＥＸＴ

［ ＰＡ／<プログラムアドレス> ］
［ ＤＡ／<データアドレス> ］
［ Ｄ／<データ> ］
［ Ｃ／<ＣＰＵステータス> ］
［ Ｅ／<外部プローブ> ］
［ Ｐ／<パスカウント> ］

(2) ＥＸＴ

［<パスカウント>＊］ＴＧｍ［－［<パスカウント>＊］ＴＧｍ　・・・］
（　ＴＧｍの論理式です　）

ｍ　　：トリガの番号（０から３）

機能　外部トリガ出力条件の設定を行います。

解説　指定した条件を外部トリガ出力条件として設定します。

プログラム実行時に外部トリガ出力条件に達すると、外部トリガに信号を出力します。

各パラメータ、およびキーワードの意味は、次の通りです。
詳細は　ＢＢ（バスブレーク設定）コマンドを参照してください。

ＰＡ	プログラムが実行するアドレス
ＤＡ	プログラムがアクセスするアドレス
Ｄ	プログラムがアクセスするデータ
Ｃ	ＣＰＵのステータス
Ｅ	外部プローブ信号
Ｐ	パスカウント,上記条件の成立回数
ＴＧｍ	トリガ条件ｍ（ｍは０から３）

注意）
・本コマンドを使用しますと２本の（ハードウエア）パスカウンタの現在の計測値は
（設定時にエラーが発生しても）０にクリヤされます。

例

１０００番地にデータ１０をライトしたときに外部トリガに信号を出力します。

```
>EXT DA/1000 D/10 C/W
```

書式　？ＥＸＴ

機能　外部トリガの出力条件の表示を行います。

解説　現在の外部トリガの出力条件の表示を行います。

　外部トリガの出力条件をファイルにセーブして、再びファイルよりリードして設定することが可能です。

例

現在の外部トリガ出力条件の表示を行います。

```
>?EXT
TGO DA=2000 C=R
BBO TGO
```

現在の外部トリガ出力条件を一時ファイルに保存し、その後再び使用します。

```
>?EXT > EXTSET.SAV       （外部トリガ出力条件をファイルに退避します。）
..................       （いろいろなコマンドを実行します。）
><EXTSET.SAV             （外部トリガ出力条件をファイルの内容よりリードし設定します。）
```

書式　＿ＥＸＴ

機能　外部トリガ出力条件の削除を行います。

解説　現在の外部トリガ出力条件を削除します。

例

現在の外部トリガ出力条件を削除します。

```
>_EXT
```

書式　(1) ＦＣＶＴ ＜ｆｌｏａｔ＞

(2) ＦＣＶＴ ＜ＨＥＸ＞ ＜ＨＥＸ＞ ［＜ＨＥＸ＞ ＜ＨＥＸ＞］

＜ｆｌｏａｔ＞：小数点が必要
＜ＨＥＸ＞　　：４桁の１６進数、２組はｆｌｏａｔ、４組はｄｏｕｂｌｅ

機能　浮動小数点数値とビット値を双方向に変換します。

解説　浮動小数点数値または、１６進数値データを与えると変換結果を表示します。リダイレクト機能によりメモリ内容の変換も可能です。

（１）ＦＣＶＴ ＜ｆｌｏａｔ＞

小数点を指定した場合、浮動小数点数値から１６進数値の変換を行います。

＜ｆｌｏａｔ＞は、次のフォーマットが可能です。

【digits】．【digits】【E|e【-|+】digits】
　　　　　~~
　　　　　小数点は省略できません。

（２）ＦＣＶＴ ＜ＨＥＸ＞ ＜ＨＥＸ＞ ［＜ＨＥＸ＞ ＜ＨＥＸ＞］

小数点を指定しない場合、ビット値から浮動小数点数値の変換を行います。

＜ＨＥＸ＞は、４桁の１６進数で２バイト（１６ビット）になります。
＜ＨＥＸ＞を２組指定した場合、計4バイトでｆｌｏａｔに変換します。
＜ＨＥＸ＞を４組指定した場合、計8バイトでｄｏｕｂｌｅに変換します。
＜ＨＥＸ＞と＜ＨＥＸ＞の間は空白を入れて下さい。

例

(1) 浮動小数点数値をビット値に変換して、メモリに書込みます。

```
>FCVT 0.1 (または1.0E-1)
float = 3DCC CCCD           (1.000000E-001)
double= 3FB9 9999 9999 999A (1.000000E-001)
>EW 200 3DCC CCCD
    ~~~ ~~~~~~~~~ アドレス200番地に3DCC、202番地にCCCDを書込みます。
```

(2) ビット値から浮動小数点数値を求めます。

```
>FCVT 3E4C CCCD
float = 0.200000(2.000000E-001)

>FCVT 3FC9 9999 9999 999A
      ~~~~ ~~~~ ~~~~ ~~~~ ４組指定した場合自動的にdoubleに変換します。
double= 0.200000(2.000000E-001)
```

FILL　　メモリ内同一パターン設定

書式

ＦＩＬＬ［／N］［／Ｂ｜／Ｗ｜／Ｌ］
＜レンジ＞　＜データ＞　［＜データ＞・・・］

Ｎ ： ベリファイなし
Ｂ ： バイトアクセスします
Ｗ ： ワードアクセスします
Ｌ ： ロングアクセスします

機能

メモリをあるデータで満たします。

解説

指定した＜レンジ＞の範囲を、＜データ＞列の値で満たします。

メモリへのライトはベリファイを行いながら実行されます。
Ｎオプションを指定しますとベリファイは行いません。

次のオプションによりアクセスサイズを指定します。

Ｂ ・・・ バイトアクセスします。（省略値）
Ｗ ・・・ ワードアクセスします。
Ｌ ・・・ ロングアクセスします。

＜データ＞には、数式のほか　’（アポストロフィ）で囲むことにより文字列も
使用できます。

例

１０００から１ＦＦＦ番地まで０クリアします。

```
>FILL 1000 1FFF 0
```

１０００から１ＦＦＦ番地まで　０、１、２、３のパターンをバイト単位で設定します。

```
>FILL 1000 1FFF 0 1 2 3
```

書式　ＦＫ　<番号>　<文字列>

<番号>　：　1から１０（ＰＣ９８０１）
1から１２（ＰＣ／ＡＴ）
<文字列>：　１５文字まで

機能　ファンクションキーの設定を行います。

解説　指定された番号のファンクションキーに文字列を定義します。

<番号>のｎは、Ｆ・ｎ に対応します。

空白は、　”　記号で囲むことにより使用することができます。
リターンキーは、　￥ｒ　の記号で表します。

<文字列>は１５文字までが有効です。

補注）
ファンクションキーの初期設定は、ＩＣＥＩＮＩＴ．ＤＡＴのＦＫコマンドで定義されます。このファイルの内容をエディタで書き換えることによりお客様の好みにあったファンクションキーの設定にすることが可能です。

例

Ｆ・１　を　ＧＯ　２０００<ＣＲ>　に定義します。

```
>FK 1 "GO 2000\r"
```

書式　？ＦＫ　［　＜番号＞　・・・　］

＜番号＞　：　１から１０（ＰＣ９８０１）
　　　　　　　１から１２（ＰＣ／ＡＴ）

機能　ファンクションキーに定義されている文字列を表示します。

解説　指定されたファンクションキーに定義されている文字列を表示します。

＜番号＞のｎは、Ｆ・ｎ に対応します。

番号を省略すると全てのファンクションキーについて表示します。

例

Ｆ・１０　に定義されている文字列を表示します。

```
>?FK 10
FK 10 "GO 2000\r"
```

書式　＿ＦＫ　＜番号＞　［　・・・　］

＜番号＞　：　１から１０（ＰＣ９８０１）
　　　　　　　１から１２（ＰＣ／ＡＴ）

機能　ファンクションキーの設定を解除します。

解説　指定された番号のファンクションキーの定義を解除します。

＜番号＞のｎは、Ｆ・ｎ に対応します。

例

Ｆ・１　を未定義にします。

```
>_FK 1
```

書式　ＧＯ　［＜アドレス＞］

機能　プログラムを実行します。

解説　指定した<アドレス>から、ユーザプログラムを実行します。
＜アドレス＞を指定しないと現在のプログラムカウンタから実行します。

補注）
・コマンドファイルの中で、ＧＯコマンドを使用後ブレークが発生する場合は、ＧＯWコマンドの使用をお勧めします。コマンドファイルを再実行したときにブレーク発生のタイミングを再現することが可能です。
・プログラムの開始位置にソフトブレークが設定されているときは１命令ステップ実行してから、プログラムを実行します。ステップ実行時、成立回数のカウント（「AT ソフトブレークポイント設定」参照）は行いません。

例

１０００番地から実行します。

```
>GO 1000
```

現在のプログラムカウンタから実行します。

```
>GO
```

書式　ＧＯＲＥＳ

機能　リセットスタートを行います。

解説　ＣＰＵにリセットを入力後、エミュレーションプログラムを実行します。

本コマンドを使用すればリセット時のデバックを行うことができます。

補注）
・コマンドファイルの中で、ＧＯＲＥＳコマンドを使用後ブレークが発生する場合は、ＲＥＳＥＴ；ＧＯWコマンドの使用をお勧めします。コマンドファイルを再実行したときにブレーク発生のタイミングを再現することが可能です。

例

ＣＰＵを初期化した後実行する場合。

```
>GORES
```

書式　ＧＯＷ　［＜アドレス＞］

機能　プログラムを実行しブレーク待ちになります。

解説　指定した<アドレス>から、ユーザプログラムを実行します。
＜アドレス＞を指定しないと現在のプログラムカウンタから実行します。

ＧＯコマンドとの違いは、ブレークが発生するまでコマンド待の状態にならないことです。
本コマンドは、コマンドファイルやマクロコマンドの中で有効に使用できます。

補注）
- ブレークが発生せずコマンド待ちにならないときはＳＴＯＰまたはCTRL+Pauseキーを入力して強制終了させてください。
- プログラムの開始位置にソフトブレークが設定されているときは１命令ステップ実行してから、プログラムを実行します。ステップ実行時、成立回数のカウント（「AT ソフトブレークポイント設定」参照）は行いません。

例

ＧＯＷコマンドにより、プログラムの実行のコマンドをロギングファイルに記録して、コマンドファイルを作成し、再び、自動的に再実行します。

```
>LOG ATGO               （ロギングを記録します。）
>AT 1000                （ブレークポイントを設定します。）
>GOW 100                （プログラムを実行します。ブレークが発生するまで、
                          待の状態になります。）
>REG                    （ブレーク発生後レジスタ値を表示します。）
>_LOG                   （ロギングの記録を解除します。）

>AUTO ATGO              （ＡＵＴＯ　コマンドで上記の手順を再実行します。）

>< ATGO.LOG             （コマンドファイルの実行により、上記の手順を再実行します。）
```

書式　(1) ＨＩＳ　［［－］数］
　　　　(2) ＨＩＳ　{ON｜OFF}

機能　コマンドの履歴を表示します。

解説　入力されたコマンドの履歴（コマンドヒストリ）を表示します。

(1) 書式 (1) に関して

ＨＩＳ　［［－］数］

パラメータにより次の機能を行います。

<数>　・・・　最近入力した数までの履歴を、キー入力順に表示します。

－<数>・・・　最近入力した数までの履歴を、キー入力の逆順に表示します。

　パラメータを省略しますと、コマンドヒストリ用のバッファにある全てのコマンドの履歴を表示します。

(2) 書式 (2) に関して

ＨＩＳ　{ON｜OFF}

パラメータにより次の機能を行います。

ON　　・・・　コマンドの履歴の取得を開始します。

OFF　　・・・　コマンドの履歴の取得を停止します。

補注）
・コマンドヒストリ用のバッファは、１Ｋバイト用意してあります。
・ＩＣＥモニタ終了時（ＱＵＩＴコマンド実行時）、モニタ起動ディレクトリのＩＣＥ．ＨＩＳファイルにその内容を退避します。ＩＣＥ立ち上げ時、同じファイルから読み込んで、コマンドの履歴を回復します。

例

最近入力した１０番目までのコマンドの履歴を表示します。
```
>HIS 10
```

書式　＿ＨＩＳ

機能　コマンドの履歴を削除します。

解説　現在まで取得されたコマンドの履歴の情報を削除します。

例

コマンドの履歴を削除します。
>_HIS

書式　(1)　ＩＣＥＳＲＣ　［<パス名>　［<パス名>］・・・］
　　　　<パス名>に'*'を指定すると前回のパス名を利用可能

機能　ソースファイルを表示する場合に参照するパス名の設定・表示を行います。

解説　ＩＣＥモニタのソースコードデバッグ機能を利用する場合、ソースファイルを参照するためソースファイルが存在するパス名を設定する必要があります。

本コマンドはＩＣＥモニタの内部コマンドとしてソースファイルの参照パス名の設定・表示を行います。これにより、一時的にソースファイルの参照パス名を追加したり、変更することが可能です。

本コマンドは環境変数"ICESRC"の内容を直接書替えることはありません。

(1)　書式(1)に関して

ＩＣＥＳＲＣ　［<パス名>　［<パス名>］・・・］

<パス名>は空白で区切って複数指定可能です。
<パス名>を省略した場合、ソースファイルの参照パス名を表示します。
<パス名>に'*'を指定すると、ICESRCの前回のパス名に置換えます。
<パス名>の記憶領域は200バイト用意してあります。

例

現在のソースファイルの参照パス名を表示する場合。

```
>ICESRC
ICESRC "A:\H8\C;A:\H8\ASM"          ;２つのパス名が設定されている例
        ~~~~~~~ ~~~~~~~~~
```

ソースファイルの参照パス名を設定した後、プログラムをロードする場合。

```
>ICESRC  B:\ICE\DEMO  C:\SRC        ;２つのパス名を設定します
         ~~~~~~~~~~~  ~~~~~~
>LOAD B:\ICE\DEMO\SAMPLE.ABS        ;『B:\ICE\DEMO\SAMPLE.C』が存在する場合
                                    ;参照可能になります。
```

ソースファイルの参照パス名を追加する場合。

```
>ICESRC  A:\H8\C  A:\H8\ASM          ;既に２つのパス名が設定されています。
>ICESRC  *  B:\ICE\DEMO              ;'*'はA:\H8\CとA:\H8\ASMに同意です。
         ~~~ ~~~~~~~~~~~             ;パス名『B:\ICE\DEMO』を３番目に追加
```

書式　Ａ＞ＩＣＥＳＹＭ　［＜オプション＞］　＜ファイル名＞

＜オプション＞
／Ｏ＜パス名＞：　デバッグ情報ファイルを書き込むパス名の指定
　　　　　　　　　省略時はカレントディレクトリに書き込みます
／Ｉ　　　　　：　デバッグ情報に関するレポートを行います
　　　　　　　　　本オプション指定時にはデバッグ情報ファイルの
　　　　　　　　　書き込みはありません

機能　リンク後のプログラムファイルからデバッグ情報ファイルを作成します。

解説　ＩＣＥモニタ上でシンボリックデバッグを行う場合、本コマンドで作成したデバッグ情報ファイル（ＩＣＥファイル）が必要になります。

(1) ＩＣＥファイルの作成

本コマンドはリンカの補助コマンドとしてお考え下さい。したがって、ロード時に毎回必要になるのではなく、リンク直後に必要になります。
アセンブル・コンパイル・リンク処理をＭＳ－ＤＯＳのバッチ処理で記述している場合は、リンク後に本コマンドを追加して下さい。
（詳細は「３．コンパイル、およびアセンブル手順」を参照してください。）

例：ユーザプログラム作成用のバッチ処理内容

```
CH38 SAMPLE1.SRC /DEBUG
ASM38 /DEBUG SAMPLE2.C
LNK SAMPLE1.OBJ,SAMPLE2.OBJ /OUT=SAMPLE.ABS /DEBUG
ICESYM SAMPLE.ABS
~~~~~~~~~~~~~~~~~バッチファイルのリンク後に追加します。
```

(2) デバッグ情報エリアの容量の設定

本コマンドの実行時にデバッグ情報量の概算をレポートするため、次のようにプロテクトモードメモリの割当て容量を決定できます。

① 本コマンドを実行します。

```
A>ICESYM <file>
ICESYM My-ICE Symbol File Generator
Copyright (C) HITACHI MICROCOMPUTER SYSTEM, LTD.
Licensed Material of HITACHI MICROCOMPUTER SYSTEM, LTD.
  Target    "<file>"
  Type      "SYSROF2"
  Optimize  "ON"
  Frametype "SP"
  File          5(1KB)
  Function     21(1KB)
  Symbol     8004(344KB)
  Auto         46(2KB)
  Line       8062(48KB)
  DSO         143(2KB)
  Inspect    3849(241KB)
  Attrib       85(1KB)
  Alloc         640KB
―――――――――――――
  Create  "<file(.ice)>"
  Total         420KB
                ~~~~~ デバッグ情報量の合計
```

② デバッグ情報量の合計値を参考にして、［コンフィグレーション］ダイアログの［ＳＹＭＢ］にシンボルバッファサイズを指定してください。

　／Ｉオプションを指定した場合、デバッグ情報量の概算を高速にレポートする事ができます。

例

リンク後デバッグ情報を作成する場合。

```
A>LNK ・・・/OUT=SAMPLE.ABS /DEBUG          ;SYSROFフォーマットファイル作成
A>ICESYM SAMPLE.ABS                         ;ＩＣＥファイルSAMPLE.ICE作成
  ~~~~~~~~~~~~~~~~~~~ＭＳ－ＤＯＳ外部コマンドとして実行します
                                            ;ＩＣＥモニタ起動
>LOAD SAMPLE                                ;SAMPLE.ABS,SAMPLE.ICEをロード
```

リンクを完了したプログラムファイルのデバッグ情報量を確認する場合。

```
A>ICESYM /I SAMPLE.ABS                      ;SAMPLE.ICEは作成しません
```

デバッグ情報ファイルの書き込み先を指定する場合。

```
A>ICESYM /OB:\SUB SAMPLE.ABS                ;B:\SUB\SAMPLE.ICEを作成します
```

書式
(1) ＩＮＳＰＥＣＴ　<式>
(2) ＩＮＳＰＥＣＴ／Ｒ

機能
　Ｃ言語の構造体変数およびポインタ・配列変数の内容をインスペクトウィンドウに登録します。

解説
(1)　Ｃ言語の変数を登録すると、その変数内容をインスペクトウィンドウ上に表示します。

(2)　すべてのインスペクトウィンドウをクローズします。

　表示内容は、エミュレーション終了後やメモリの変更後に自動的に再表示されるため、変数内容を効率的に監視できます。

　［Repaint］ボタンにより、インスペクトウィンドウを再表示することも可能です。

　<式>には、Ｃ言語の変数や構造体をはじめ配列表現やキャストの指定が可能ですが、複雑な指定には括弧を明示する必要があります。

　ｉｎｔ型とビット数が等しい整数型はｉｎｔ型として表示します。
(SHシリーズのlong，unsigned long，SH以外のshort，unsigned short)

　インスペクトウィンドウの登録は最大５０個まで可能です。
　インスペクトウィンドウ内の操作は「4.4.18 インスペクトウィンドウ」を参照してください。

例

Ｃ言語の構造体 tag_data を登録する場合。
```
>INSPECT tag_data
```

ｌｏｎｇ型の変数をｃｈａｒ型にキャストして登録する場合。
```
>INSPECT ((char [4])&dword_data)
```

アドレスＡ０００番地を構造体 xdata_t 型にキャストして登録する場合。
```
>INSPECT (*(struct xdata_t *)0xA000)
```

書式

ＬＩＮＥ　＜信号種別＞／＜モード＞　　・・・
　　＜信号種別＞：ＢＲＥＱ ＵＳＥＲ ＲＥＳ
　　＜モード＞　：Ｅ Ｄ

機能

外部信号の制御（許可、禁止等）の設定をします。

解説

エミュレーション実行時、ＣＰＵに対する指定された＜信号種別＞の信号の制御を指定された＜モード＞に設定します。

<信号種別>としては、次のものが指定できます。

ＢＲＥＱ	ＢＲＥＱの入力
ＵＳＥＲ	エミュレーションＲＡＭアクセス時の ＷＡＩＴ，ＡＳ，ＤＳ，ＲＤ，ＷR信号の入出力
ＲＥＳ	リセットの入力

＜信号種別＞に対し<モード＞としては、次のものが指定できます。

Ｅ	許可する
Ｄ	禁止する（マスクする）

例

ＢＲＥＱの入力を禁止します。

```
>LINE BREQ/D
```

書式　？ＬＩＮＥ

機能　外部信号の制御（許可、禁止等）の状態を表示します。

解説　エミュレーション実行時、ＣＰＵに対する信号の制御状態を表示します。

ファイルに退避した外部信号制御情報をリードして、再び、同一状態に設定することができます。
例を参照してください。

例

すべての外部信号の入力制御状態を表示します。
>?LINE

一度外部信号の入力制御状態をファイルに退避して、その後、再び、待避した外部信号の入力制御状態をリードして同一条件に制御を設定します。

```
>?LINE > LINE.SAV       （現在の外部信号の入力制御状態をファイルに退避します。）
  ・・・                 （いろいろな、エミュレーションコマンドを実行します。）
>< LINE.SAV             （先に退避した外部信号の入力制御状態をファイルよりリードして、
                          外部信号の入力制御の設定を行います。）
```

書式　ＬＯＡＤ［<オプション>］ <ファイル名>

<オプション>
／Ｙ　：　ＳＹＳＲＯＦフォーマット（省略値）
／Ｇ　：　ＳＹＳＲＯＦフォーマット
（変数のシンボルをすべてグローバルとしてロード）
／Ｓ　：　Ｓタイプフォーマット
／Ｂ　：　バイナリフォーマット
／Ｃ　：　プログラムのみをロードします
／Ｄ　：　シンボルのみの登録

機能　プログラムのロードを行います。

解説　指定したファイルからプログラムをロードします。

(1) オプションによりロードモジュールのタイプを次のように指定します。

Ｙ　ＳＹＳＲＯＦタイプフォーマット
リロケータブルモジュールはロードできません。
拡張子を省略した場合、”．ＡＢＳ”を付加します。
シンボル情報はＩＣＥファイル（拡張子は．ＩＣＥ）から
読み込みます。
ＩＣＥファイルの作成手順は、マニュアルの『ICESYM』
を参照下さい。

Ｇ　Ｙオプションと同じですが変数のシンボルをすべて
グローバルとして登録します。

Ｓ　モトローラＳタイプフォーマットロードモジュール

Ｂ　バイナリデータロードモジュール
ＳＡＶＥコマンドで作成したバイナリデータモジュー
ルをロードします。

Ｃ　Ｙオプションと同じですがプログラムのみをロードし、
シンボル情報の登録は行いません。

Ｄ　Ｙオプションと同じですがシンボル情報のみを登録し、
プログラムのロードは行いません。

(2) ＩＣＥモニタのシンボル情報エリアの容量は、［コンフィグレーション］
ダイアログの［ＳＹＭＢ］で指定できます。

(3) ソースコードデバッグを行うには、コンパイル・アセンブル・リンク時に
デバッグ情報作成オプションを指定してください。
また、ＩＣＥＳＹＭコマンドによりＩＣＥファイルを作成して下さい。
（詳細は「３．コンパイル、およびアセンブル手順」を参照してください。）

日立製Ｃコンパイラ・アセンブラ（／ｄｅｂｕｇ）

```
A>shc  /debug <file>
A>asmsh <file> /debug
A>lnk <file> /debug
A>icesym <file>
```

ＬＯＡＤコマンドを実行すると、ロード前に登録されていたシンボルを削除します。

例

ＳＹＳＲＯＦフォーマットのロードモジュールをロードする場合。

>LOAD PROGRAM1	PROGRAM1.ABS と PROGRAM1.ICE を読み込みます。 エラーメッセージ『シンボルファイル(.ICE)が見つかりません』 を表示する場合、ＩＣＥＳＹＭコマンドによりＩＣＥファイルを 作成して再実行して下さい。 例: >ICESYM PROGRAM1 　　>LOAD PROGRAM1 ＩＣＥＳＹＭコマンドはリンカの補助コマンドとしてお考え下さい。 したがって、リンク後に必ず実行するようにして下さい。

変数のシンボルをすべてグローバルとして登録する場合。

>LOAD/G SAMPLE	アセンブラ記述時のデバッグに効果があります。 .EXPORT を指定しないシンボルに対してファイル名を指定せずに参照 可能になります。

セーブしたプログラムをロードする場合。

>SAVE/S P1 1000 1FFF	（モトローラＳタイプフォーマットでセーブ）
>LOAD/S P1	（モトローラＳタイプフォーマットでロード）
>SAVE/B P2 2000 2FFF	（バイナリフォーマットでセーブ）
>LOAD/B P2	（バイナリフォーマットでロード）

書式　ＬＯＣＡＬ

機能　Ｃコンパイラのデバッグ情報から自動変数の内容をコマンドウィンドウに表示します。

解説　Ｃ言語で記述した関数内を実行中に、自動変数の内容を表示します。

　オプションウィンドウのローカルシンボル表示では、本コマンドと同じ内容をオプションウィンドウのサイズ内で表示します。

　Ｃ言語の自動変数は、一般にスタックエリアに割当てられ、関数の始めと終りの割当て・解放部分では正しい内容を表示しない事がありますので注意してください。

例

関数 func_sub() を８ステップ実行後、自動変数の内容を確認する場合。

```
>LOAD SAMPLE
>AT %func_sub
>GO
>STEP 8
>LOCAL
```

書式　ＬＯＧ　＜ファイル名＞

機能　エミュレータのコマンドや表示結果をファイルへ記録します。

解説　キー入力や、その表示結果をロギング情報としてファイル（ロギングファイル）に保存します。

ファイル名には拡張子を指定しないでください。

ロギングを取りますと　ＡＵＴＯ　コマンドで自動再テストを行うことができます。
例を参照してください。

注意）
　＜ファイル名＞に拡張子　．ＯＵＴ　を付加したファイルにコマンド入力と実行結果を記録します。
　＜ファイル名＞に拡張子　．ＬＯＧ　を付加したファイルにコマンド入力を記録します。

例

一度実行したコマンドを　ロギングファイル　に記録し再び自動的に再実行します。

```
>LOG LOGSAV            （ロギングを取得します。ファイル名に拡張子は付けません。）
..........             （いろいろなＩＣＥのコマンドを実行します）
>_LOG                  （ロギングの取得を終了します。）
..........
>CAT  LOGSAV.OUT       （先ほどの実行結果を表示します。）
..........
>AUTO LOGSAV           （自動再テストを実行します。）
```

書式　？ＬＯＧ

機能　ロギング中のファイル名を表示します。

解説　現在ロギング中であれば、 ロギングファイル名を表示します。
ロギング中でなければなにも表示しません。

例

```
>LOG LOGSAV              (ロギングを取得します。)
..........               (いろいろなＩＣＥのコマンドを実行します)
>?LOG                    (ロギングファイル名を表示します。)
LOG LOGSAV
```

_LOG　　　　**ロギング情報の取得終了**

書式　　**＿ＬＯＧ**

機能　　**ロギングの取得を終了します。**

解説　　**ロギングの取得を終了します。**

例

ロギングの取得を終了します。
```
>_LOG
```

MAP　　マッピング

書式　ＭＡＰ　＜レンジ＞　＜モード＞

（１）ＭＡＰ　＜レンジ＞　{ＥＲＡＭ|ＵＲＡＭ|ＷＡＴＣＨ}

（２）ＭＡＰ　＜レンジ＞　{ＷＲＰＲＯ|　ＧＵＡＲＤ|ＮＯＢＲＫ}

機能　メモリ空間のマッピングを行います。

解説　メモリ空間のマッピングの設定を行います。

(1) エミュレーションメモリおよびＷＡＴＣＨメモリの割り当て（書式１）

外部空間のとき<モード>として次のものが４Ｋバイト単位で使用できます。

ＥＲＡＭ	エミュレーションメモリを割り付けます。
ＵＲＡＭ	ユーザ空間とします。
ＷＡＴＣＨ	ＷＡＴＣＨメモリを割り付けます。

(2) ＭＡＰブレークの設定（書式２）

全空間に<モード>として次のものが８Ｋバイト単位で使用できます。

ＷＲＰＲＯ	ライトプロテクトエリアに指定します。
ＧＵＡＲＤ	ガードエリアに指定します。
ＮＯＢＲＫ	ＭＡＰブレーク指定を解除します。

プログラムがライトプロテクトエリアにライトしたとき、プログラムがガードエリアをアクセスしたとき、次のようなメッセージを表示します。

ＷＲＰＲＯの場合	WRITE PROTECTED ！
ＧＵＡＲＤの場合	GOURD ！

ＷＡＴＣＨメモリはエミュレーションメモリを使用します。
エミュレーションメモリを使い切ると次のようなメッセージが表示されています。
エミュレーションメモリの容量はステータスウィンドウに表示されます。

「エミュレーションメモリが不足しています」

例

０番地から１ＦＦＦＦ番地までエミュレーションメモリを割り当てます。

```
>MAP 0 1FFFF ERAM
```

書式　？ＭＡＰ　［｛＜レンジ＞｜＜アドレス＞｝］

機能　メモリ空間のマッピングの表示を行います。

解説　メモリ空間のマッピングの状態を表示します。

＜レンジ＞を指定しますと、指定した空間のマッピングの状態を表示します。
＜アドレス＞を指定しますと、指定したアドレスのマッピングの状態を表示します。
パラメータを省略しますと、全空間のマッピングの状態を表示します。

空間のモードの意味は、ＭＡＰコマンドと同じです。

ファイルに退避したマッピング情報をリードして、再び、同一状態にマッピングすることができます。例を参照してください。

例

現在のマッピングの状態を表示します。

```
>?MAP
```

一度マッピングの状態をファイルに退避して、その後、再び、退避したマッピングの状態をリードして同一条件でマッピングを設定します。

```
>?MAP > MAP.SAV        （現在のマッピング状態をファイルに退避します。）
 .....                 （いろいろな、エミュレーションコマンドを実行します。）
>< MAP.SAV             （先に退避したマッピングの状態をファイルよりリードして、
                         マッピングの設定を行います。）
```

書式　＿ＭＡＰ

機能　空間のマッピングを解除します。

解説　空間のマッピングを解除して初期状態に戻します。

例

空間のマッピングを解除して初期状態に戻します。

```
>_MAP
```

書式　**ＭＯＶＥ　<レンジ>　<開始アドレス>**

機能　**メモリのある範囲のデータを移動します。**

解説　**指定した<レンジ>の範囲のデータを、<開始アドレス>から始まる位置に移動します。**

例

１０００番地から１ＦＦＦ番地までの内容を３０００番地からに移動します。

```
>MOVE 1000 1FFF 3000
```

書式

ＯＮ　＜ＯＮ条件＞
＜ＯＮ条件＞

ＡＴ	:	ソフトブレーク条件成立時
ＰＢ	:	ＰＣブレーク条件成立時
ＢＢｎ	:	バスブレークｎ成立時
ＳＴＯＰ	:	ＳＴＯＰコマンド終了時
ＳＭＰＬ	:	サンプリングタイマ条件成立時

機能

ＯＮ条件が成立した時の処理を定義します。

解説

指定した＜ＯＮ条件＞が成立した時に実行する処理を定義します。

実行する処理はマクロコマンドとして登録されます。
＜ＯＮ条件＞には次のものがあります。括弧内に定義されるマクロコマンド名を示します。

ＡＴ	ソフトブレーク条件成立時（ＯＮＡＴ）
ＰＢ	ＰＣブレーク条件ｎ成立時（ＯＮＰＢ）
ＢＢｎ	バスブレークｎ成立時（ＯＮＢＢｎ）
	ｎ：　０　１　２　３
ＳＴＯＰ	ＳＴＯＰコマンド終了時（ＯＮＳＴＯＰ）
ＳＭＰＬ	サンプリングタイマ条件成立時（ＯＮＳＭＰＬ）

これらの初期値（ＳＭＰＬを除く）は現在のレジスタ値の表示と、現在のプログラムカウンタが指す命令の逆アセンブル表示となっています。
これらの処理の登録方法はマクロコマンドの登録と同じです。

補注）
・次のようなマクロコマンドの登録と同じ意味を持ちます。
ＣＭＤ　ＯＮ＜ＯＮ条件＞
例）
”ＯＮ　ＳＴＯＰ”　コマンドと　”ＣＭＤ　ＯＮＳＴＯＰ”
は同じ意味を持ちます。

例

ソフトブレーク成立時　Ｒ０レジスタ　の指すメモリ内容をダンプするようにします。

```
>ON AT
? DUMP .R0
?.
macro_name.
>
```

書式　？ＯＮ　［<ＯＮ条件>］

<ＯＮ条件>

ＡＴ	:	ソフトブレーク条件成立時
ＰＢ	:	ＰＣブレーク条件成立時
ＢＢｎ	:	バスブレークｎ成立時
ＳＴＯＰ	:	ＳＴＯＰコマンド終了時
ＳＭＰＬ	:	サンプリングタイマ条件成立時

機能　ＯＮ条件が成立した時の処理を表示します。

解説　指定した<ＯＮ条件>が成立した時に実行する処理を表示します。

実行する処理はマクロコマンドとして登録されてます。
<ＯＮ条件>には次のものがあります。括弧内に定義されるマクロコマンド名を示します。

ＡＴ	ソフトブレーク条件成立時（ＯＮＡＴ）
ＰＢ	ＰＣブレーク条件ｎ成立時（ＯＮＰＢ）
ＢＢｎ	バスブレークｎ成立時（ＯＮＢＢｎ） ｎ：　０　１　２　３
ＳＴＯＰ	ＳＴＯＰコマンド終了時（ＯＮＳＴＯＰ）
ＳＭＰＬ	サンプリングタイマ条件成立時（ＯＮＳＭＰＬ）

これらの初期値（ＳＭＰＬを除く）は現在のレジスタ値の表示と、現在のプログラムカウンタが指す命令の逆アセンブル表示となっています。
これらの処理の表示内容はマクロコマンドの表示と同じです。

補注）
・次のようなマクロコマンドの表示と同じ意味を持ちます。
　？ＣＭＤ　ＯＮ<ＯＮ条件>
　例）”？ＯＮ　ＳＴＯＰ”コマンドと”？ＣＭＤ　ＯＮＳＴＯＰ”
　　は同じ意味を持ちます。

例

ソフトブレーク成立時の処理を表示します。

```
>?ON AT
ONAT
        ECHO "BREAK AT !\n"
        REG
        DASM  .PC .PC
.
.
```

書式　(1)　ＰＡＳＳ
　　　　　［ ＰＡ／<プログラムアドレス> ］
　　　　　［ ＤＡ／<データアドレス> ］
　　　　　［ Ｄ／<データ> ］
　　　　　［ Ｃ／<ＣＰＵステータス> ］
　　　　　［ Ｅ／<外部プローブ> ］

(2)　ＰＡＳＳ　　ＴＧｍ

ｍ　：トリガの番号（０から３）

機能　パスカウント測定条件の設定を行います。

解説　指定した条件をパスカウント測定条件として設定します。

プログラム実行時にパスカウント測定条件に達すると、パスカウント（通過回数）をカウントします。

各パラメータ、およびキーワードの意味は、次の通りです。
詳細は　ＢＢ（バスブレーク設定）コマンドを参照してください。

ＰＡ	プログラムが実行するアドレス
ＤＡ	プログラムがアクセスするアドレス
Ｄ	プログラムがアクセスするデータ
Ｃ	ＣＰＵのステータス
Ｅ	外部プローブ信号
ＴＧｍ	トリガ条件ｍ（ｍは０から３）

パスカウンタは、エミュレーションプログラム実行のコマンド（ＧＯ、ＧＯＷ、ＧＯＲＥＳ）でクリアされます。

注意）

・本コマンドを使用しますと２本の（ハードウエア）パスカウンタの現在の計測値は（設定時にエラーが発生しても）０にクリヤされます。

例

プログラムが１０００番地を何回通過したかを計測します。

```
>PASS PA/1000
```

書式　？ＰＡＳＳ

機能　パスカウント測定条件とパスカウントの表示を行います。

解説　パスカウント測定条件と現在のパスカウント（通過回数）の表示を行います。

補注）
・パスカウントの計測は、リアルタイムで行われます。
・リアルタイムパラレルモードで本コマンドを実行することが可能です。
・パスカウントが６５５３６以上の場合オーバーフローで表示されます。

例

パスカウント測定条件とパスカウントの表示を行います。

```
>?PASS
TGO PA=H'001000 C=R PF
PASS TGO
REM 5742回成立
```

書式　＿ＰＡＳＳ

機能　パスカウント測定条件の削除を行います。

解説　現在のパスカウント測定条件を削除します。

例

現在のパスカウント測定条件を削除します。
```
>_PASS
```

書式　ＰＢｎ　　＜アドレス＞［、＜回数＞］

ｎ　　　　　　：ＰＢの番号（０から３）

機能　ＰＣブレークポイントの設定を行います。

解説　指定したアドレスにＰＣブレークポイントを設定します。
プログラム実行時ブレークポイントに達すると、ブレークポイントの命令を実行後停止します。

　ブレークポイントは、ＰＢ０からＰＢ３の４個登録できます。
　<回数>は、ブレークポイントが成立するまでの回数を指定します。
<回数>を指定しないと、パスカウント=1が採られます。＜回数＞は最大６５５３５回です。
　成立回数は、プログラムによりカウントされます。

実行停止後、ＯＮ　ＰＢコマンドで定義された処理（ＯＮＰＢマクロコマンド）が実行されます。
ＯＮ　ＰＢコマンドの初期値は実行後のレジスタの値と次に実行する命令を表示するようになっています。

ＰＣブレークはハードウェアによるブレーク機能なので、ユーザ実機のＲＯＭ空間にも使用できます。

ブレーク条件をファイルにセーブして、その後再びファイルよりリードして、同じブレーク条件を設定できます。

例

関数名ｔｔｓｕｂに　ＰＣブレークを設定します。
```
>PB0 %ttsub
```

現在のＰＣ＋２番地を６回実行するとＰＣブレークします。
```
>PB0 .PC+2,6
```

一度ＰＣブレーク条件をファイルに退避して、再び、ファイルに退避したブレーク条件を設定します。
```
>?PB1 > BRKAT.SAV      （ソフトブレーク条件をファイルにセーブ）
 ................      （いろいろなコマンドを実行）
>< BRKAT.SAV           （ファイルにセーブしたソフトブレーク条件を設定します）
```

書式　？ＰＢ［ｎ］

ｎ　　　：ＰＢの番号（０から３）

機能　ＰＣブレークポイントの表示を行います。

解説　指定したブレークポイントの条件の表示を行います。

ｎ（数字）を指定しますと　ＰＢｎ　に設定されているブレークポイントを表示します。
ｎ（数字）を省略しますと　ＰＢ０　から　ＰＢ３　に設定されているブレークポイントを表示します。

ブレーク条件をファイルにセーブして、その後再びファイルよりリードして、同じブレーク条件を設定できます。

例

全てのＰＣブレークポイントを表示します。

```
>?PB
PB0  H'010000,1 H'002000,1
PB1  1000,20
```

ＰＣブレーク条件をファイルにセーブします。

```
>?PB > AT10.SAV
```

書式　＿ＰＢ［ｎ］　［<アドレス>］

　　ｎ　　　　：ＰＢの番号（０から３）

機能　ＰＣブレークポイントの削除を行います。

解説　指定したソフトブレークポイントの条件の削除を行います。

＿ＰＢ　と入力しますと全てのＰＣブレークを削除します。

例

１０００番地にＰＣブレークポイントを設定しその後削除します。

```
>PB0 1000
  ..........
>_PB0 1000
```

書式　(1) ＰＥＲ　Ｍ［ｎ］　＜レンジ＞　［{ＣＯＮＴ｜ＯＮＥ}］

ｎ　　：　０から７
ＳＥＱ　：　サブルーチン等の実行時間含む
ＥＸ　　：　サブルーチン等の実行時間含まない
ＣＯＮＴ：　継続計測
ＯＮＥ　：　最初の１回のみ計測

(2) ＰＥＲ　Ｉ　＜レンジ＞　＜最小時間＞　＜最大時間＞［{ＢＲＫＬ｜ＢＲＫＨ}］

時間　　：　＜秒＞Ｓ＜ミリ秒＞Ｍ＜マイクロ秒＞Ｕ
ＳＥＱ　：　サブルーチン等の実行時間含む
ＥＸ　　：　サブルーチン等の実行時間含まない
ＢＲＫＬ：　最小時間より短いときブレーク
ＢＲＫＨ：　最大時間より長いときブレーク

(3) ＰＥＲ　ＯＮ

(4) ＰＥＲ　ＤＭＰ／［ＴＩＭＥ｜ＣＯＵＮＴ｜ＩＮＴ］

機能　パフォーマンス測定条件の設定を行います。

解説　コマンドのパラメータにより次のようなパフォーマンス（実行時間）測定条件の設定、および測定結果の表示を行います。

（１）ＰＥＲ　Ｍ［ｎ］　＜レンジ＞［{ＳＥＱ｜ＥＸ}　{ＣＯＮＴ｜ＯＮＥ}］

実行時間、および実行回数を測定するモジュールの範囲を登録します。

ｎは、登録したモジュールを区別するためのモジュール番号で０から７までの数です。同時に８個のモジュールのパフォーマンスの測定を行うとができます。モジュール０は、下記の（２）で述べるような目的にも使用されますので、モジュール１からモジュール７を使用することをお勧めします。

ｎを省略したときは、モジュール１からモジュール７、最後にモジュール０の順で空きとなっているモジュールを割当ます。

＜レンジ＞でモジュールの範囲を指定します。

例）　１０００　　１ＦＦＦ　・・・　１０００番地から１ＦＦＦ番地まで
　　　％ＳＴＡＲＴ　￥２００・・・　ＳＴＡＲＴ番地から２００バイト分

ＣＯＮＴ、ＯＮＥ、で測定のモードを指定します。

ＳＥＱ　・・・　モジュールの先頭番地を実行してから、最終番地を実行するまでの時間を測定します。（省略値）

ＥＸ　・・・　モジュールの範囲を実行した時間を測定します。モジュールがサブルーチンを実行している時間や割り込み処理に要してる時間等を、排除して実行時間を測定します。

ＣＯＮＴ　・・・　指定したモジュールが実行される毎に実行時間を加算します。（省略値）

ＯＮＥ　・・・　指定したモジュールが最初の一回実行されたときのみ実行時間を測定します。

例）

```
PER  M1 %SUB1 %SUB2-1 CONT
```

本コマンドによりモジュール登録後、ＰＥＲ ＤＭＰ／ＴＩＭＥまたはＰＥＲ ＤＭＰ／ＣＯＵＮＴコマンドにより、プログラム全体の実行時間にしめるモジュールの実行時間や実行回数の割合（百分率）を表示することができます。

（２）ＰＥＲ　Ｉ　<レンジ><最小時間>　<最大時間>
　　　　　　　　　　　　　　　　　　［ＳＥＱ｜ＥＸ］　［{ＢＲＫＬ｜ＢＲＫＨ}］

あるモジュールの実行時間分布測定の指定をおこないます。モジュールは、<レンジ>で指定します。
ここで指定したモジュールは、上記（１）で説明したモジュール０に自動的に割当られます。

実行時間の分布のインターバル時間は<最小時間>と<最大時間>の間を等分した値（小数点以下切捨て）になります。実行時間の分布は、この等分された時間の区間と、<最小時間>より小さい時間と、<最大時間>より大きい時間について測定されます。

<最小時間>と<最大時間>の指定方法は次の通りです。

<秒の値>Ｓ<ミリ秒の値>Ｍ<マイクロ秒の値>Ｕ
例）１Ｓ２Ｍ３Ｕ　（1秒２ミリ秒３マイクロ秒）

ＳＥＱ，ＥＸで測定のモードを指定します。

ＳＥＱ　・・・　モジュールの先頭番地を実行してから、最終番地を実行するまでの時間を測定します。（省略値）
ＥＸ　・・・　モジュールの範囲を実行した時間を測定します。モジュールがサブルーチンを実行している時間や割り込み処理に要してる時間等を、排除して実行時間を測定します。

例）
```
PER I %SUB1 %SUB2-1  1S2M3U  4S5M6U
```

本コマンドによりモジュール登録後、ＰＥＲ　ＤＭＰ／ＩＮＴコマンドにより、モジュールの実行時間分布を表示することができます。

ＢＲＫＨを指定すると<最大時間>より大きいときにブレークします。
ＢＲＫＬを指定すると<最小時間>より小さいときにブレークします。

注意）

パフォーマンスブレーク後、プログラムを再実行したりステップ実行したりする前に「ＰＥＲ ＯＮ」を実行してください。

（３）ＰＥＲ　ＯＮ

実行時間測定の開始を指定します。

ＯＮ　・・・　実行時間測定を開始します。

上記（１）、（２）の設定を行ったときは、本コマンドを実行してください。

ＰＥＲ　ＯＮコマンド実行時それまで測定したパファーマンス情報もクリアされます。

（４）ＰＥＲ　ＤＭＰ／［ＴＩＭＥ｜ＣＯＵＮＴ｜ＩＮＴ］

実行時間の測定結果をコマンドウィンドウに表示します。

ＴＩＭＥ　・・・（１）で指定したモジュール群の実行時間と百分率を表示します。
ＣＯＵＮＴ　・・・（１）で指定したモジュール群の実行回数と百分率を表示します。
ＩＮＴ　・・・（２）で指定したモジュール内の実行時間分布と百分率を表示します。
これらは、先頭の一文字（Ｔ、Ｃ、Ｉ）でも指定可能です。

例

サブルーチン　ＳＵＢＸ、ＳＵＢＹ、ＳＵＢＺの全体の実行時間に占める割合を表示します。
サブルーチン　ＳＵＢＸ、ＳＵＢＹ、ＳＵＢＺ、ＳＵＢWはメモリ上に連続して配置されており、
各サブルーチンの範囲は次の値（シンボル）の通りとします。

ＳＵＢＸ　・・・　ＳＵＢＸ　から　ＳＵＢＹ－１　まで
ＳＵＢＹ　・・・　ＳＵＢＹ　から　ＳＵＢＺ－１　まで
ＳＵＢＺ　・・・　ＳＵＢＺ　から　ＳＵＢＷ－１　まで

```
>PER M %SUBX %SUBY-1
>PER M %SUBY %SUBZ-1
>PER M %SUBZ %SUBW-1
>PER ON
>GO
>STOP
>PER DMP/TIME
```

書式　？ＰＥＲ

機能　パフォーマンス測定条件の表示を行います。

解説　現在設定されている全てのパフォーマンスの測定条件の表示を行います。

例

パフォーマンスの測定条件を表示します。

```
>?PER
PER M1  H'010000 H'01FFFF
PER M2  H'002000 H'0020FF
```

書式　＿ＰＥＲ　　　［｛ｎ｜Ｉ｝］

　　ｎ　　　：０から７

機能　パフォーマンス測定条件の削除を行います。

解説　指定したパフォーマンス測定条件の削除を行います。

パラメータにより次のようにパフォーマンスの測定条件を削除します。

ｎ　　・・・　指定したモジュールの区間のパフォーマンス測定条件を削除します。

Ｉ　　・・・　実行時間分布の測定条件を解除します。

パラメータを指定しませんと設定されている全てのパフォーマンス条件を削除します。

例

１０００番地と２ＦＦＦ番地にパフォーマンス測定条件を設定しその後削除します。

```
>PER  M1  1000 2FFF
  ..........
>_PER 1
```

書式

(1) ＰＳＴＥＰ　［＜　カウンタ　＞］
(2) ＰＳＴＥＰ　＜プログラムアドレス＞　＜カウンタ＞

機能

サブルーチンコール命令は１命令としてステップ実行します。

解説

ＢＳＲ、ＪＳＲ、およびトラップ命令は１命令としてステップ実行します。

(1) 書式(1)に関して
現在のプログラムカウンタから指定した＜カウンタ＞数分の命令を実行します。
＜カウンタ＞を指定しないと、１命令のみ、実行します。

(2) 書式(2)に関して
指定した＜プログラムアドレス＞から＜カウンタ＞数分の命令を実行します。

＜カウンタ＞は、基数を省略した場合１０進数がとられます。

例

４命令分関数ステップします。

```
>PSTEP 4
```

書式　ＱＵＩＴ

機能　エミュレータプログラムを終了します。

解説　エミュレータプログラムを終了します。

例

```
>QUIT
```

REG　レジスタ値の表示

書式　ＲＥＧ　［　＜レジスタ名＞　・・・　］

＜レジスタ名＞：
ＰＣ　ＣＣＲ
Ｒ０　Ｒ１　Ｒ２　Ｒ３　Ｒ４　Ｒ５　Ｒ６　Ｒ７
Ｒ０Ｌ　Ｒ１Ｌ　Ｒ２Ｌ　Ｒ３Ｌ　Ｒ４Ｌ　Ｒ５Ｌ　Ｒ６Ｌ　Ｒ７Ｌ
Ｒ０Ｈ　Ｒ１Ｈ　Ｒ２Ｈ　Ｒ３Ｈ　Ｒ４Ｈ　Ｒ５Ｈ　Ｒ６Ｈ　Ｒ７Ｈ
Ｅ０　Ｅ１　Ｅ２　Ｅ３　Ｅ４　Ｅ５　Ｅ６　Ｅ７
ＥＲ０　ＥＲ１　ＥＲ２　ＥＲ３　ＥＲ４　ＥＲ５　ＥＲ６　ＥＲ７

機能　レジスタの値の表示を行います。

解説　指定されたレジスタの値を表示します。

〈レジスタ名〉を省略すると、全てのレジスタ値を表示します。
〈レジスタ名〉を指定するとそのレジスタの値を表示します。
〈レジスタ名〉として次が使用できます。

ＰＣ　ＣＣＲ
Ｒ０　Ｒ１　Ｒ２　Ｒ３　Ｒ４　Ｒ５　Ｒ６　Ｒ７
Ｒ０Ｌ　Ｒ１Ｌ　Ｒ２Ｌ　Ｒ３Ｌ　Ｒ４Ｌ　Ｒ５Ｌ　Ｒ６Ｌ　Ｒ７Ｌ
Ｒ０Ｈ　Ｒ１Ｈ　Ｒ２Ｈ　Ｒ３Ｈ　Ｒ４Ｈ　Ｒ５Ｈ　Ｒ６Ｈ　Ｒ７Ｈ
Ｅ０　Ｅ１　Ｅ２　Ｅ３　Ｅ４　Ｅ５　Ｅ６　Ｅ７
ＥＲ０　ＥＲ１　ＥＲ２　ＥＲ３　ＥＲ４　ＥＲ５　ＥＲ６　ＥＲ７

レジスタの値を一時ファイルに退避して、その後再び、退避したファイルの内容をリードしレジスタに設定することができます。例を参照してください。

例

全レジスタを表示します。
```
>REG
```

レジスタ　ＰＣ　Ｒ０　Ｒ３　を表示します。
```
>REG PC R0 R3
```

レジスタの値を一時ファイルに退避し、その後、再び退避したファイルの内容をリードしてレジスタに設定します。

```
>REG > ALLREGS          （全レジスタの値をファイルに退避します。）
...............         （いろいろなコマンドを実行します。）
>< ALLREGS              （先ほど退避したレジスタの値をファイルよりリードして、
                          現在のレジスタにセットします。）
```

書式　ＲＥＭ　　　［＜文字列＞］

機能　コメントの行です。

解説　コメントの行として扱い、何も実行しません。

コマンドファイル内にコメント用として使用しますと便利です。

補注）
　文字列の中に　；　は使用できません。

例

```
>REM   EXECUTE GO
>GO
>REM   THIS IS COMMENT ！
```

書式　ＲＥＳＥＴ　［<アドレス>］

<アドレス>　：　プログラムカウンタに設定するアドレス

機能　ＣＰＵを初期化します。

解説　ＣＰＵに対してリセットを入力し、ＣＰＵの内部状態と内蔵周辺モジュールの各レジスタを初期化します。

<アドレス>の設定を省略した場合、メモリ上のベクタ内容からリセットＰＣ初期値を設定します。

例

ＣＰＵを初期化して、プログラムカウンタをＨ’１００番地に設定する場合。

```
>RESET 100
```

書式　ＲＥＴＵＲＮ

機能　Ｃ言語で記述した関数内から関数の呼出し元まで戻ります。

解説　Ｃ言語で記述した関数の戻りアドレスに対してブレークの設定を行い、自動的にＧＯコマンドを実行します。

関数の途中から残りの部分を一気に実行するので、リターンコードの確認などのテストに有効です。

戻りアドレスは、一般にスタックエリアに格納されているため、関数の初めと終わりの部分では、正しく動作しない場合があります。
バックトレースで戻りアドレスが表示されるか確認してください。

例

テスト中の関数から呼出し元まで戻る場合。

```
>RETURN
```

書式　ＳＡＶＥ［｛／Ｓ｜／Ｂ｝］　＜ファイル名＞　＜レンジ＞

Ｓ　：　Ｓタイプフォーマット（省略値）
Ｂ　：　バイナリフォーマット

機能　プログラムの保存を行います。

解説　指定した<ファイル名>にプログラムを保存します。

オプションによりロードモジュールのタイプを次のように指定します。

Ｓ　Ｓタイプフォーマットロードモジュール
（省略値）

Ｂ　バイナリデータロードモジュール

例

２０００から３０００番地をＰＲＯＧ１．ＳＡＶに保存します。

```
>SAVE PROG1.SAV 2000 3000
```

２０００から５０００バイト分をＰＲＯＧ２．ＳＡＶに保存します。

```
>SAVE PROG2.SAV 2000 \5000
```

書式　ＳＥＴ　＜環境種別＞／＜モード＞　　・・・

＜環境種別＞／＜モード＞
　　ＣＬＯＣＫ／｛Ｕ｜Ｕ２｜Ｏ｜Ｏ２｜Ｏ４｜Ｏ８｜
　　　　　１｜２｜４｜８｜１０｜１３｜１６｜１７｜１８｝
　　ＭＯＤＥ　／｛１｜２｜３｜４｜５｜６｜７｝
　　ＴＡＢ／コードウィンドウ内でのタブスキップのサイズ
　　ＳＴＥＰ／｛ＳＲＣ｜ＭＩＸ|WND｝

機能　エミュレータの動作環境の設定を行います。

解説　エミュレータの動作環境の設定を行います。

＜環境種別＞には次のものが指定できます。

ＣＬＯＣＫ	クロックのスピード
ＭＯＤＥ	ＣＰＵのモード
ＴＡＢ	タブサイズ
ＳＴＥＰ	ステップ実行のモード

＜モード＞には次のものが指定できます。

（１）ＣＬＯＣＫ
エミュレーションＣＰＵのクロックを指定します。

Ｕ	外部のクロックを使用する。
Ｕ２	外部のクロックを２分周して使用する。
Ｕ４	外部のクロックを４分周して使用する。
Ｕ８	外部のクロックを８分周して使用する。
Ｏ	発振子のクロックを使用する。
Ｏ２	発振子のクロックを２分周して使用する。
Ｏ４	発振子のクロックを４分周して使用する。
Ｏ８	発振子のクロックを８分周して使用する。
１	１ＭＨｚのシステムクロックを使用する。
２	２ＭＨｚのシステムクロックを使用する。
４	４ＭＨｚのシステムクロックを使用する。
８	８ＭＨｚのシステムクロックを使用する。（初期値）
１０	１０ＭＨｚのシステムクロックを使用する。
１３	１３ＭＨｚのシステムクロックを使用する。
１６	１６ＭＨｚのシステムクロックを使用する。
１７	１７ＭＨｚのシステムクロックを使用する。
１８	１８ＭＨｚのシステムクロックを使用する。

注）・デバイスの最大周波数を超えるクロックは選択しないでください。
　・“Ｏ”、“Ｏ２”、“Ｏ４”、“Ｏ８”はＨ８／３０５２，３０２２，３０２１，３０２０では指定できません。
　・“Ｕ４”、“Ｕ８”はＨ８／３０５２，３０２２，３０２１，３０２０以外では指定できません。

（２）ＭＯＤＥ
ＣＰＵのモードを指定します。
初期値はモード１です。

（４）ＴＡＢ
コードウィンドウ内にソースプログラムを表示するとき、タブコードに対してタブスキップするサイズ（１から８０）を指定します。

（5）ＳＴＥＰ

ＳＲＣ	ソース行単位でステップ実行を行います。
ＭＩＸ	機械語単位でステップ実行を行います。
WND	ソースウィンドウが選択されているときはソース行単位で、混在モードウィンドウが選択されているときは機械語単位でステップ実行します。

例

外部のクロックを使用し、モード１に設定します。

```
>SET CLOCK/U  MODE/1
```

書式　？ＳＥＴ

機能　エミュレータの動作環境の状態を表示します。

解説　現在のエミュレータの動作環境の状態を表示します。

ファイルに退避したエミュレータの動作環境情報をリードして、再び、
同一状態の動作環境に設定することができます。例を参照してください。

例

現在のエミュレータの動作環境を表示します。
>?SET
一度エミュレーションの動作環境をファイルに退避して、その後再び、
退避した動作環境をリードして同一条件に制御を設定します。
>?SET> SET.SAV　　（現在のエミュレーションの動作環境をファイルに退避します。）
　・・・　　　　　　（いろいろな、エミュレーションコマンドを実行します。）
>< SET.SAV　　　　（先に退避した動作環境をファイルよりリードして、動作環境の
　　　　　　　　　　設定を行います。）

書式　ＳＭＰＬ　　<式（ｍｓｅｃ）>

機能　サンプリングタイマをスタートします。

解説　サンプリングタイマをスタートします。

指定した時間に達すると　ＯＮ　ＳＭＰＬ　コマンド　で定義した（ＯＮＳＭＰＬマクロコマンド）コマンドを実行します。

補注）
- ・サンプリングタイマには、２５０ｍｓｅｃのタイマ割り込みを使用してします。
- ・基数は１０進数がとられます。
- ・最大値は、２１４７４８３６４７です。
- ・ＯＮＳＭＰＬマクロコマンド実行中サンプリングタイマは一時停止します。

例

５秒毎に１０００番地の内容を表示します。本処理はリアルタイムパラレルモードで実行されるためメモリは内蔵ＲＯＭ、内蔵ＲＡＭ、およびエミュレーションラムでなけれはなりません。

```
>SMPL 5000                    (サンプリングタイマTM0を
                               5000msecに設定します。)
>ON SMPL
?LOCATE 0                     (カーソルを先頭に移動します。)
?DUMP 1000                    (リアルタイムパラレルモードで1000番地を表示します。)
?.
macro_name?.
>
```

書式　？ＳＭＰＬ

機能　サンプリングタイマの設定値を表示します。

解説　指定したサンプリングタイマ条件の設定値を表示します。
単位はｍｓｅｃです。

例

サンプリングタイマの値を表示します。

```
>?SMPL
```

書式　＿ＳＭＰＬ

機能　サンプリングタイマを削除します。

解説　指定したサンプリングタイマ条件を削除します。

例

サンプリングタイマを削除します。

```
>_SMPL
```

書式　ＳＲＣＨ ＜レンジ＞　＜データ＞　［　＜データ＞　・・・　］

機能　メモリのある範囲のデータをサーチします。

解説　指定した<レンジ>の範囲のデータ内で、<データ>と一致するものを捜します。

サーチするデータ長は指定したデータの個数により決まります。＜データ＞を１個指定したときは１バイトのデータを、２個指定した時は２バイトのデータを、ｎ個指定した時はｎバイトのデータをサーチします。

文字列を指定したいときは　’（アポストロフィ）で文字列を囲みます。

例

１０００番地から１００バイト分　０　のデータをサーチします。

```
>SRCH 1000  \100  0
FOUND
   001000   001001   001004   001005   001008   001009   00100B   00100F
   001010   001011   001014   001015   001016   001017   001018   001019
   00101A   00101B   00101C   00101D   00101E   00101F   001020   001021
   001022   001023   001024   001025   001026   001027   001028   001029
```

文字列　１２３４　をサーチします。

```
>SRCH 20000 \80 '1234'
```

書式　ＳＲＥＧ　［<レジスタ名>［／<式>　・・・］］

<レジスタ名>：
ＰＣ　ＣＣＲ
Ｒ０　Ｒ１　Ｒ２　Ｒ３　Ｒ４　Ｒ５　Ｒ６　Ｒ７
Ｒ０Ｌ　Ｒ１Ｌ　Ｒ２Ｌ　Ｒ３Ｌ　Ｒ４Ｌ　Ｒ５Ｌ　Ｒ６Ｌ　Ｒ７Ｌ
Ｒ０Ｈ　Ｒ１Ｈ　Ｒ２Ｈ　Ｒ３Ｈ　Ｒ４Ｈ　Ｒ５Ｈ　Ｒ６Ｈ　Ｒ７Ｈ
Ｅ０　Ｅ１　Ｅ２　Ｅ３　Ｅ４　Ｅ５　Ｅ６　Ｅ７
ＥＲ０　ＥＲ１　ＥＲ２　ＥＲ３　ＥＲ４　ＥＲ５　ＥＲ６　ＥＲ７

機能　レジスタの値の変更を行います。

解説　指定したレジスタの値を変更します。
<レジスタ名>として次が使用できます。

ＰＣ　ＣＣＲ
Ｒ０　Ｒ１　Ｒ２　Ｒ３　Ｒ４　Ｒ５　Ｒ６　Ｒ７
Ｒ０Ｌ　Ｒ１Ｌ　Ｒ２Ｌ　Ｒ３Ｌ　Ｒ４Ｌ　Ｒ５Ｌ　Ｒ６Ｌ　Ｒ７Ｌ
Ｒ０Ｈ　Ｒ１Ｈ　Ｒ２Ｈ　Ｒ３Ｈ　Ｒ４Ｈ　Ｒ５Ｈ　Ｒ６Ｈ　Ｒ７Ｈ
Ｅ０　Ｅ１　Ｅ２　Ｅ３　Ｅ４　Ｅ５　Ｅ６　Ｅ７
ＥＲ０　ＥＲ１　ＥＲ２　ＥＲ３　ＥＲ４　ＥＲ５　ＥＲ６　ＥＲ７

パラメータを省略しますと、現在のレジスタの値を表示してからレジスタの値を変更することができます。例を参照してください。

例

全レジスタの値を参照しながら変更します。

```
>SREG
PC    000000 > %start          (レジスタ名と現在の値が表示されますので設定したい値を指定します。)
.......                        (以下同様に設定可能です。)
ER0 00000123 >                 (キャリッジリターンのみ入力しますと、レジスタの値は変更せずに
                                次のレジスタの値を表示します。)
ER1 00000123 > 89
ER2 00000123 > ^               (^を入力しますと1つ前に戻ります。)
ER1 00000089 > .               (.を入力しますと終了します。)
>
```

Ｒ１レジスタの値を参照しながら変更します。

```
>SREG  R1
R1  0123 > 89
R2  0123 > .
>
```

プログラムカウンタと、Ｒ０レジスタの値を変更します。

```
>SREG  PC/%start R0/10
```

レジスタの値を一時ファイルに退避し、その後、再び退避したファイルの内容をリードしてレジスタに設定します。

```
>REG > ALLREGS         (全レジスタの値をファイルに退避します。)
...............        (いろいろなコマンドを実行します。)
>< ALLREGS             (先ほど退避したレジスタの値をファイルよりリードして、
                        現在のレジスタにセットします。)
```

書式

(1) ＳＴＥＰ　＜　カウンタ　＞]
(2) ＳＴＥＰ　＜プログラムアドレス＞　＜カウンタ＞

機能

１命令ごとに、プログラムを実行します。

解説

１命令ごとに、プログラムを実行します。

(1) 書式(1)に関して
現在のプログラムカウンタから指定した＜カウンタ＞数分の命令を実行します。
＜カウンタ＞を指定しないと、１命令のみ、実行します。

(2) 書式(2)に関して
指定した＜プログラムアドレス＞から＜カウンタ＞数分の命令を実行します。

＜カウンタ＞は、基数を省略した場合１０進数がとられます。

例

１命令実行します。

```
>STEP
```

書式　ＳＴＯＰ

機能　プログラムを中断します。

解説　実行中のエミュレーションプログラムを中断します。

　ＳＴＯＰ命令実行後、ＯＮ　ＳＴＯＰ　コマンドで定義された処理（ＯＮＳＴＯＰ　マクロコマンド）が実行されます。初期値は実行後のレジスタの値と次に実行する命令を表示するようになっています。

例

エミュレーション実行を中断します。

```
>STOP
```

書式

ＳＵＢＳ ［｛／Ｂ｜／Ｗ｜／Ｌ｝］［／Ｎ］
　　　　＜　アドレス　＞［＜　データ　＞　・・・］

Ｂ ： １バイト単位でライト
Ｗ ： ２バイト単位でライト
Ｌ ： ４バイト単位でライト
Ｎ ： ベリファイなし

機能

メモリの内容を書き換えます。

解説

指定した、＜アドレス＞の内容を＜データ＞で書き換えます。

＜データ＞を省略しますと、指定した＜アドレス＞のデータの内容を表示しますので順次メモリの内容を変更することが可能です。例を参照してください。
’　で囲むことにより文字列の設定が可能です。
オプションにより次の機能を選択できます。

Ｎ	書き込み後、データを読み込んでのベリファイチェックは行いません。
Ｂ	１バイト単位で変更します。（省略値）
Ｗ	２バイト単位で変更します。
Ｌ	４バイト単位で変更します。

補注）
内蔵レジスタの領域を指定したときベリファイは行いません。

例

メモリの内容を参照しながら変更します。

```
>SUBS 1000
001000 23 > 00          （番地と現在の内容が表示されますので設定したい値を指定します。）
.......                 （以下同様に設定可能です。）
001003 11 >             （キャリッジリターンのみ入力しますと、メモリの内容は変更せずに
                          次の番地の内容を表示します。）
001004 00 > '1234'      （文字列を設定します。）
001009 22 > 89
00100A 33 > ^           （＾を入力しますと１つ前に戻ります。）
001009 89 > .           （. を入力しますと終了します。）
>
```

１０００番地からの内容を　００　０１　０２　に設定します。

```
>SUBS/B 1000  0 1 2
```

Ｒ０レジスタが指す番地に、Ｒ１レジスタの値を設定します。

```
>SUBS/W .R0 .R1
```

シンボル　ＤＡＴＡ　に　’１２３４５６７８’　を設定します。

```
>SUBS %DATA '12345678'
```

SYM　　　シンボル値登録

書式　ＳＹＭ ［｛＜シンボル名＞　＜式＞・・・｜＜　＜ファイル名＞　］

機能　シンボルに式の値を代入します。

解説　指定したシンボルに式の値を代入します。

シンボル名がすでに存在しているときは、そのシンボルの値が変更されます。
新しいシンボル名を指定したときは新規にシンボルを登録して式の値を代入します。

シンボル名は、英字、？、＿　で始まる　英数字、？、＿　の文字列で、最大３２文字までです。
シンボルはグローバル変数として扱われます。
パラメータを省略しますと、　ＳＹＭ＞　のプロンプトが表示されますので、シンボルを連続して登録することが可能です。例を参照してください。

例

現在のプログラムカウンタの値をシンボル名　ＳＵＢ１　に割り当てます。

```
>SYM  SUB1  .PC
```

シンボル名　ＳＴＡＲＴＡ　に　ＦＦＦ　加えた値を、シンボル名　ＥＤＮＡ　に割り当てます。

```
>SYM ENDA %STARTA+FFF
```

シンボルを連続して登録します。

```
>SYM                          （パラメータなしでＳＹＭコマンドを実行します。）
SYM>SYM1 1234 SYM2 2345       （シンボルを登録します。）
SYM>SYM4 4444                 （シンボルの登録を続行します。）
SYM>                          （キャリッジリターンのみ入力すると、
                               ＳＹＭコマンドを終了します。）
>
```

書式　？ＳＹＭ　［＜シンボル名＞　・・・　］

機能　シンボルの値の表示を行います。

解説　指定したシンボルの値を表示します。

＜シンボル名＞を入力しますと、指定したシンボル名の値を表示します。
＜シンボル名＞を省略しますと、現在登録されている全てのシンボル名とその値を表示します。

ファイルに現在のシンボル（グローバル変数）の値を退避して、その後再びファイルより読み込んでシンボル（グローバル変数）の値を設定することができます。例を参照してください。

例

全てのシンボルの値を表示します。

```
>?SYM
;     ＜グローバル変数＞
count          0000042E        xval           00000430
mem            00000446
;     ＜スタティック変数＞
USERV2.C:seed  00000434        USERV2.C:msg0  00000436
USERV2.C:msg7  00000444
;     ＜グローバル関数＞
main           00000000        v_calc         000000EE
v_check        00000134        sub_0          00000159
;     ＜スタティック関数＞
```

現在のシンボルの値を一度ファイルに退避してその後、再びファイルの内容をリードして同じシンボルの値に設定します。

```
>?SYM > SYMFILE.SAV     （現在のシンボルの値をファイルに退避します。）
>Q                      （エミュレータを終了します。）

>SYM < SYMFILE.SAV      （シンボルの値を先に退避したファイルの内容より
                          リードして設定します。）
```

書式　＿ＳＹＭ

機能　シンボルの削除を行います。

解説　現在登録されている全てのシンボル名を削除します。

例

全てのシンボルを削除します。

```
>_SYM
```

書式　ＴＤＭＰ［｛／Ｍ｜／Ｌ｜／Ｄ｝］
　　　　［<先頭フレーム番号>［<最終フレーム番号>］］
　　　　　　Ｍ　：　混在形式
　　　　　　Ｌ　：　逆アセンブル形式
　　　　　　Ｄ　：　ダンプ形式

機能　トレース情報の表示を行います。

解説　トレースメモリ内に取得されているトレース情報を表示します。

オプションにより表示形式を選択します。
　Ｍ　　混在形式で表示します。（省略値）
　Ｌ　　逆アセンブル形式で表示します。
　Ｄ　　ダンプ形式で表示します。

フレーム番号は、バスサイクル単位で番号がふられ最後の　バスサイクルが、フレーム番号　１となり、最古のフレーム番号が最も大きいフレーム番号をとります。
<先頭フレーム番号>から<最終フレーム番号>までのトレースメモリの内容を表示します。
<最終フレーム番号>を省略したときは、<先頭フレーム番号>から、１２インストラクション分のトレースメモリの内容を表示します。
<先頭フレーム番号>、<最終フレーム番号>を共に省略したときは、全てのトレースメモリの内容を表示します。

(1) 混在形式

混在形式での表示形式を下図に示します。

```
    +---フレーム番号
    |   +---トレースON、OFF
    |   |   +---アドレス
    |   |   |    +---データ
    |   |   |    |     +---アクセス空間
    |   |   |    |     |    +---CPUステータス
    |   |   |    |     |    |    +---IACK信号
    |   |   |    |     |    |    |   +---クロック数
    |   |   |    |     |    |    |   |
 f-no ton addr  data  ma   st  iack clk
 00167    000000 01   EX8  PF        008
      $  main:
      $   user2.c:0019:main()          /* メイン関数 */
      *  000000 01 00 6D F6             PUSH.L        ER6
 00166    000001   00 EX8  PF        006
 00165    000002 6D   EX8  PF        006
 00164    000003   F6 EX8  PF        006
 00163    000004 0F   EX8  PF        006
 00162    000005   F6 EX8  PF        006
 00161    0FFF0C 0000 IRAM W         004
      *  000004 0F F6                   MOV.L         ER7,ER6
 00160    0FFF0E 0000 IRAM W         002
 00159    000006 01   EX8  PF        006
      *  000006 01 00 6D F2             PUSH.L        ER2
 00158    000007   00 EX8  PF        006
 00157    000008 6D   EX8  PF        006
 00156    000009   F2 EX8  PF        006
 00155    00000A 7A   EX8  PF        006
```

・ソースプログラム表示
実行サイクル時、実行アドレスがソースプログラム行の
アドレスと一致したとき，そのソースファイル名と行番号,
ソース行を表示します（先頭に　$　を表示します）。

・アセンブル表示
実行サイクル時、命令を逆アセンブル表示します。
（先頭に　＊　を表示します）

・クロック数
バスサイクル中のクロック数を表示します。

・アドレス
アドレスバスの値を表示します。

・データ
データバスの値を表示します。バイトアクセス時には
１バイト、ワードアクセス時には２バイト表示します。

・アクセス空間
アクセス空間の種別を表示します。

ＩＲＯＭ	内蔵ＲＯＭ空間をアクセス
ＩＲＡＭ	内蔵ＲＡＭ空間をアクセス
ＩＯ８	8ビットバス内蔵レジスタ空間をアクセス
ＩＯ１６	16ビットバス内蔵レジスタ空間をアクセス
ＥＸ８	8ビットバス外部メモリ空間をアクセス
ＥＸ１６	16ビットバス外部メモリ空間をアクセス

・ＣＰＵステータス
ＣＰＵの動作状態を表示します。

ＰＦ	プリフェッチ
Ｒ	データリード
Ｗ	データライト
ＤＭＡＲ	ＤＭＡリード
ＤＭＡＷ	ＤＭＡライト
ＲＥＦ	リフレッシュサイクル

・ＩＡＣＫ
例外処理の受付

・外部プローブ信号

(2) 逆アセンブル形式

逆アセンブル形式での表示形式を下図に示します。

```
+---フレーム番号
   |         +---アドレス
   |         |     +---データ               +---アセンブル命令
   |         |     |                        |
f-no       addr  data                  mnemonic
     $   USER2.C:0016:        for( p = mem, i = count; ; i++ ){
00038*   00004E 7A 05 00 00 05 66       MOV.L         #00000566:32,ER5
00035*   000054 01 00 6F E5 FF D0       MOV.L         ER5,@(FFD0:16,ER6)
```

(3) ダンプ形式

ダンプ形式での表示形式を下図に示します。

```
f-no ton addr   data  ma   st  iack clk
00012   00008A 01   EX8  PF          026
00011   00008B   00 EX8  PF          006
```

補注）
ＡＴ、ＰＢコマンドでパスカウントを指定したときは、最後の一つ前のブレーク発生から、最後のブレーク発生までの内容がトレースされます。

例

全実行命令を表示します。
>TDMP

書式

ＴＧ［ｎ］
［ＰＡ／＜プログラムアドレス＞］
［ＤＡ／＜データアドレス＞　　］
［Ｄ／＜データ＞　　　　　　　］
［Ｃ／＜ＣＰＵステータス＞　　］
［Ｅ／＜外部プローブ＞　　　　］

ｎ　　　　　　　　：トリガの番号（０から３）
＜ＣＰＵステータス＞　：Ｒ，Ｗ，ＲＷ，ＰＦ，ＩＡＣＫ
ＷＯＲＤ，ＢＹＴＥ，ＤＭＡ，ＲＥＦ
ＩＲＯＭ，ＩＲＡＭ，ＩＲＥＧ，ＥＸＴ

機能

トリガの設定を行います。（テーブルへの登録のみです。）

解説

指定した条件をハードウェアトリガ条件として登録します。
ここで登録したハードウェアトリガ条件を使用してシーケンシャルブレークや、
バスブレーク条件等を設定できます。

トリガ条件はは、最大４個まで登録できます（０から３）。
ｎ　は、登録するトリガの番号です。ｎを指定しないと、最初に使用可能なトリガの
番号を採ります。

各キーワードの意味は、次の通りです。

ＰＡ　　プログラムの実行アドレス（プリフェッチ時）

ＤＡ　　プログラムがアクセスするアドレス

Ｄ　　　プログラムがアクセスするデータ

Ｃ　　　ＣＰＵのステータス
次のいずれかが指定できます。

Ｒ	データのリード時
Ｗ	データのライト時
ＲＷ	データのリードまたはライト
ＷＯＲＤ	ワードアクセス
ＢＹＴＥ	バイトアクセス
ＤＭＡ	ＤＭＡサイクル
ＲＥＦ	リフレッシュサイクル
ＩＡＣＫ	例外処理の受付
ＩＲＯＭ	内蔵ＲＯＭをアクセス
ＩＲＡＭ	内蔵ＲＡＭをアクセス
ＩＲＥＧ	内蔵レジスタをアクセス
ＥＸＴ	外部メモリをアクセス

これらは次のように連結することができます。

Ｒ　ＷＯＲＤ	ワードでリード
Ｗ　ＢＹＴＥ	バイトでライト
ＲＷ　ＷＯＲＤ	ワードでリード／ライト

Ｅ　　　外部信号プローブの値
１バイトの値です。

キーワード　ＰＡ、ＤＡ、Ｄには、ドントケアの指定を行うことができます。

例）

ＤＡ／Ｈ’１ＸＸＸ　　１６進数の１０００番台をアクセスしたらブレーク
ＤＡ／Ｏ’１ＸＸＸ　　８進数の１０００番台をアクセスしたらブレーク
ＤＡ／Ｂ’１ＸＸＸ　　２進数の１０００番台をアクセスしたらブレーク

一時トリガ条件をファイルに退避して、その後再びファイルより退避したトリガ条件を読み込み設定することができます。

例

関数ＳＵＢ１　を実行してから　関数ＳＵＢ２　を実行したときにシーケンシャルブレークします。

```
>TG0 PA/%SUB1
>TG1 PA/%SUB2
>BB0 TG0-TG1
```

一時トリガ条件をファイルに退避して、その後再びファイルより退避したトリガ条件を読み込み設定することができます。

```
>?TG1 >TRGX.SAV          （現在のトリガ条件をファイルにセーブします。）
.............            （いろいろなエミュレーションコマンドを実行します。）
><TRGX.SAV               （ファイルよりトリガ条件をリードして再び設定します。）
```

書式　？ＴＧ［ｎ］

ｎ　　　：数字（０から３）

機能　トリガ条件の表示を行います。

解説　指定したトリガ条件の表示を行います。

ｎを指定しませんと全てのトリガ条件を表示します。

例

全てのトリガ条件を表示します。

```
>?TG
TG0 DA=2000 D=20 C=W
TG1 DA=1000
```

書式　＿ＴＧ［ｎ］
　　　　　ｎ　：数字（０から３）

機能　トリガ条件の削除を行います。

解説　指定したトリガ条件の削除を行います。

ｎを指定しませんと全てのトリガ条件を削除します。

補注）
　ＴＧｎが、ＢＢｎ、ＴＩＭ、ＥＸＴ、ＴＲＣ、ＰＡＳＳにより使用中であるときは、
　削除できません。

例

１番目のトリガ条件を削除します。

```
>_TG1
```

書式

(1)　ＴＩＭ　ＧＯ
(2)　ＴＩＭ　［ＯＮ　<トリガ条件>］　［ＯＦＦ　<トリガ条件>］
　　　　　　［｛ＣＯＮＴ｜ＯＮＥ｝］

<トリガ条件>のフォーマット

(a)　［ＰＡ／<プログラムアドレス>］
　　［ＤＡ／<データアドレス>］
　　［Ｄ／<データ>］
　　［Ｃ／<ＣＰＵステータス>］
　　［Ｅ／<外部プローブ>］
　　［Ｐ／<パスカウント>］

(b)［<パスカウント>＊］ＴＧｍ［－［<パスカウント>＊］ＴＧｍ・・・］
　　（　ＴＧｍの論理式です　）

ｍ　　：トリガの番号（０から３）

機能

タイマの起動と停止条件を設定します。

解説

指定した条件をタイマの起動または停止条件として登録します。
本タイマによりエミュレーションプログラムの実行時間を測定できます。
パラメータにより起動または停止条件を指定します。

ＧＯ	ユーザプログラム実行と同時にタイマが起動します。（初期値）
ＯＮ	タイマの起動条件を指定します。
ＯＦＦ	タイマの停止条件を指定します。
ＣＯＮＴ	実行時間を加算して計測します。（省略値）
ＯＮＥ	最初の一回実行のみ実行時間を測定します。

各パラメータ、およびキーワードの意味は、次の通りです。
詳細は　ＢＢ（バスブレーク設定）コマンドを参照してください。

ＰＡ	プログラムが実行するアドレス（プリフェッチ時）
ＤＡ	プログラムがアクセスするアドレス
Ｄ	プログラムがアクセスするデータ
Ｃ	ＣＰＵのステータス
Ｅ	外部プローブ信号
Ｐ	パスカウント,上記条件の成立回数
ＴＧｍ	トリガ条件ｍ（ｍは０から３）

タイマの起動、停止条件をファイルにセーブして、再びファイルよりリードして設定することが可能です。
タイマはＧＯコマンド実行時にクリアされます。

注意）
・本コマンドを使用しますと２本の（ハードウエア）パスカウンタと現在の計測値は（設定時にエラーが発生しても）０にクリヤされます。

例

２０００番地をしたあとタイマを起動（時間の計測開始）し、３０００番地を実行するとタイマを停止（時間の計測停止）します。

```
>TIM    ON PA/2000  OFF PA/3000
```

書式　？ＴＩＭ

機能　タイマの条件とタイマの値を表示します。

解説　現在のタイマの起動、停止条件とタイマの値を表示します。

タイマの値は次のように日数、時間、分、秒、ミリ秒、マイクロ秒で表示されます。

```
2 DAYS 4H 43M 23S 842m 962u
```

日数、時間、分は、０の場合には省略します。

例

現在のタイマの設定とタイマ値を表示します。

```
>?TIM
TIM GO
REM TIMER = 20S 187m 582u
```

現在のタイマの起動、停止条件を一時ファイルに保存し、その後再び使用します。

```
>?TIM > TIMSET.SAV      （タイマの起動、停止条件をファイルに退避します。）
..................      （いろいろなコマンドを実行します。）
>< TIMSET.SAV           （タイマの起動、停止条件をファイルの内容よりリードし
                         設定します。）
```

書式　＿ＴＩＭ

機能　タイマの条件を削除します。

解説　現在のタイマの条件を削除します。

例

現在のタイマ条件を削除します。

```
>_TIM
```

書式
(1) ＴＲＣ　ＡＬＬ　　＜トレース制御＞
(2) ＴＲＣ　ＳＭＰ　　＜トリガ条件＞　＜トレース制御＞
(3) ＴＲＣ　ＲＮＧ
　　ＯＮ＜トリガ条件＞　ＯＦＦ＜トリガ条件＞　ＳＴＰ＜トリガ条件＞
　　＜トレース制御＞

＜トリガ条件＞のフォーマット
(a)　［ ＰＡ／＜プログラムアドレス＞ ］
　　［ ＤＡ／＜データアドレス＞ ］
　　［ Ｄ／＜データ＞ ］
　　［ Ｃ／＜ＣＰＵステータス＞ ］
　　［ Ｅ／＜外部プローブ＞ ］
　　［ Ｐ／＜パスカウント＞ ］
(b)［＜パスカウント＞＊］ＴＧｍ［－［＜パスカウント＞＊］ＴＧｍ　・・・］
　　（　ＴＧｍの論理式です　）
　　ｍ　：トリガの番号（０から３）

＜トレース制御＞のフォーマット
ＤＬＹ／＜サイクル数＞　ＦＵＬ／｛ＣＮＴ｜ＳＴＰ｝　ＥＮＤ／｛ＣＮＴ｜ＢＲＫ｝
ＰＲＯＢＥ

機能　トレース条件の設定を行います。

解説　指定した条件をトレース条件として設定します。

プログラム実行時に＜トリガ条件＞で指定したトレース取得条件に達すると、
トレースの取得を＜トレース制御＞に従って開始します。

(1) 書式（１）に関して

ＴＲＣ　ＡＬＬ　＜トレース制御＞

第一パラメータに　ＡＬＬ　を指定しますと、エミュレーションプログラム実行と
同時にトレース情報の取得を開始します。

(2) 書式（２）に関して

ＴＲＣ　ＳＭＰ　＜トリガ条件＞　＜トレース制御＞

第一パラメータに　ＳＭＰ　を指定しますと、＜トリガ条件＞が成立したときから
トレース情報の取得を開始します。

(3) 書式（３）に関して
ＴＲＣ　ＲＮＧ
［ＯＮ＜トリガ条件＞］　［ＯＦＦ＜トリガ条件＞］
［ＳＴＰ＜トリガ条件＞］　＜トレース制御＞

第一パラメータに、ＲＮＧを指定しますと、トレース情報の取得、中断、および停止の
トリガ条件をそれぞれ次のように指定できます。

ＯＮ　　トレース取得の開始条件を指定します。

ＯＦＦ　トレース取得の中断条件を指定します。開始条件に達すると
　　　　再びトレースの取得を行います。

ＳＴＰ　トレース取得の停止条件を指定します。開始条件に達しても
　　　　トレースの取得は行いません。

(4) ＜トリガ条件＞の説明
トリガ条件の各パラメータ、およびキーワードの意味は、次の通りです。詳細はＢＢ（バスブレーク設定）コマンドを参照してください。

ＰＡ　プログラムが実行するアドレス
ＤＡ　プログラムがアクセスするアドレス
Ｄ　プログラムがアクセスするデータ
Ｃ　ＣＰＵステータス
Ｅ　外部プローブ信号
Ｐ　パスカウント，上記条件の成立回数

ＴＧm　トリガ条件m（mは０から３）

(5) ＜トレース制御＞の説明
トレースの取得は、次のように＜トレース制御＞によりコントロールされます。

ＤＬＹ ／〈サイクル数〉
トレース取得の中断および停止条件成立後、トレースの取得を続行するバスサイクル数を指定します。０を指定すると、トレース取得の中断および停止条件成立後、直ちにトレースの取得を中断、および停止します。
省略値は、０となっています。
サイクル数は最大６５５３５まで指定できます。
ＳＭＰ、ＲＮＧのみで意味を持ちます。

ＦＵＬ／｛ＣＮＴ｜ＳＴＰ｝
トレースバッファが一杯になったときの処理を指定します。

ＣＮＴ（省略値）
トレース取得を続行します。トレースバッファはサイクリックに使用され、最も古いトレース情報が消去されて行きます。

ＳＴＰ
トレース取得を停止します。

ＥＮＤ／｛ＣＮＴ｜ＢＲＫ｝
トレース取得を停止したときの処理を指定します。

ＣＮＴ
ユーザプログラムの実行をブレークせず続行します。

ＢＲＫ（省略値）
ユーザプログラムの実行をブレークします。

ＰＲＯＢＥ

外部プローブの値をトレース取得します。ＴＤＭＰコマンドで外部プローブの値が表示されるようになります。
通常はクロック数を取得します。

一時トレース条件をファイルに退避して、その後再びファイルより退避したトレース条件を読み込み設定することができます。例を参照してください。

注意）

・本コマンドを使用しますと２本の（ハードウェア）パスカウンタの現在の測定値は（設定時にエラーが発生しても）０クリヤされます。

例

常にトレースを取得します。トレースバッファフルでエミュレーションプログラムをブレークします。

```
>TRC ALL
```

常にトレースを取得します。トレースバッファフルでも、トレース情報をサイクリックに取得し、エミュレーションプログラムを続行します。

```
>TRC ALL  FUL/CNT
```

１０００番地に１０をライトしたときにトレースします。

```
>TRC SMP DA/1000 D/10 C/W
```

関数ｓｕｂ１実行時に取得を開始し、関数ｓｕｂ２を実行したときにトレースの所得を一時中断します。（次に再びｓｕｂ１を実行しますとトレースの取得を開始します。）ｓｕｂ３を実行したときトレースの取得は停止します。

```
>TRC RNG ON   PA/%sub1 -
>        OFF  PA/%sub2 -
>        STP  PA/%sub3
```

一時トレース条件をファイルに退避して、その後再びファイルより退避したトレース条件を読み込み設定することができます。

```
>?TRC >TRACEX.SAV          （現在のトレース条件をファイルにセーブします。）
.............              （いろいろなエミュレーションコマンドを実　行します。）
><TRACEX.SAV               （ファイルよりトレース条件をリードして再び設定します。）
```

書式　？ＴＲＣ

機能　トレース条件の表示を行います。

解説　現在のトレースの条件の表示を行います。

トレース条件はファイルにセーブし再び設定することが可能です。

例

現在設定されているトレース条件を表示します。

```
>?TRC
TGO PA=H'000600 C=R PF
TRC      SMP TGO   -
         DLY/0  FUL/BRK
```

書式　＿ＴＲＣ

機能　トレース条件の削除を行います。

解説　現在設定されているトレース条件を無効とします。

例

現在設定されているトレース条件を削除します。

```
>_TRC
```

書式

(1) ＶＩＥＷ　＜アドレス＞
(2) ＶＩＥＷ　％［＜ファイル名＞：］＜行番号＞

＜アドレス＞には、％＜シンボル名＞を指定できます。

機能

ソースファイルをコードウィンドウ上に表示します。

解説

(1)　デバッグ情報がある場合は、関数名やラベル名を指定して、対応するソースファイルを表示できます。

＜シンボル名＞を省略した場合は、カレントファイル上の次のページを表示します。

(2)　ＶＩＥＷコマンドは、テキストエディタと同様に任意のファイルのリードが可能です。

＜ファイル名＞：を省略した場合は、カレントファイル上の行番号とします。

例

関数 main() のソースプログラムを表示する場合。

```
>VIEW %main
```

ソースファイル SAMPLE.C を表示する場合。

```
>VIEW %SAMPLE.C:1        (SAMPLE.C の先頭を表示します。)
>VIEW %240               (SAMPLE.C の２４０行目を表示します。)
```

書式　ＶＲＦＹ［｛／Ｙ｜／Ｓ｜／Ｂ｝］　<ファイル名>

Ｙ　：　ＳＹＳＲＯＦフォーマット（省略値）
Ｓ　：　Ｓタイプフォーマット
Ｂ　：　バイナリフォーマット

機能　プログラムの内容の比較を行います。

解説　指定したファイルのプログラムを読み込みエミュレーションＲＡＭまたはユーザメモリの内容と比較します。

オプションによりロードモジュールのタイプを次のように指定します。

Ｙ　ＳＹＳＲＯＦタイプフォーマットロードモジュール
（省略値）

Ｂ　バイナリデータロードモジュール

Ｓ　モトローラＳタイプフォーマットロードモジュール

不一致の箇所があれば、画面にそのアドレスと、メモリ内容、ロードモジュールの内容を表示します。

例

メモリ内容とファイル名ＰＲＯＧＲＡＭの内容を比較します。

```
>VRFY PROGRAM
```

書式　WATCH［／W｜／L｜／A｜／F｜／D｜／I］　<アドレス>　［<個数>］

（省略時）	：	１バイト単位に表示	＜個数＞＝１６まで
W	：	２バイト単位に表示	＜個数＞＝８まで
L	：	４バイト単位に表示	＜個数＞＝４まで
F	：	FLOATの精度で表示	＜個数＞＝１まで
D	：	DOUBLEの精度で表示	＜個数＞＝１まで
A	：	文字列の表示	＜個数＞＝６４まで
I	：	エミュレーション実行を中断して表示	

機能　ウォッチ機能を行うアドレスをウォッチウィンドウに登録します。

解説　ウォッチアドレスを登録すると、そのメモリ内容をウォッチウィンドウ上に表示します。

表示内容は、エミュレーション終了後やメモリの変更後に自動的に再表示されるため、メモリ内容を効率的に監視できます。
画面リフレッシュ（ＣＴＲＬ－））により、ウォッチウィンドウを再表示する事も可能です。

＜個数＞は、表示単位の個数で最大値は次の通りです。

１バイト単位の表示	＜個数＞＝１６まで
２バイト単位の表示	＜個数＞＝８まで
４バイト単位の表示	＜個数＞＝４まで
FLOATの精度で表示	＜個数＞＝１まで
DOUBLEの精度で表示	＜個数＞＝１まで
文字列の表示	＜個数＞＝６４まで

＜個数＞を省略した場合には、最大値とします。

ウォッチウィンドウの登録は、最大２０行まで可能です。
Ｉオプションによりウォッチウィンドの表示時に一時エミュレーションを中断してメモリの内容を参照可能です。エミュレーション中に内蔵レジスタや実機上のメモリを参照したいときに有効ですがエミュレーションの中断が絶えず行われます。また、？ＴＩＭコマンドでのタイマの値も正しく行われません。
Ｓオプションによりエミュレーションの中断を行う時間間隔を２５０ｍｓｅｃの倍数で指定できます。例を参照してください。

例

シンボル名 mem_buf から６４バイトを１バイト単位で表示する場合。
```
>WATCH %mem_buf
```

アドレスＨ’ＦＣ００から２バイト単位で４組表示する場合。
```
>WATCH/W FC00 4
```

内蔵レジスタやユーザ実機上のメモリを参照する場合です。

>W FE80	（エミュレーション実行中はリードが不可能の為、ブレーク発生後ウィンドウ内のデータの値が書替えられます。）
>W/I FF80	（エミュレーション中であれは、一時中断してメモリの内容をリードしてウィンドウ内のデータの値を絶えず書替えます。ウィンドウ内にＩオプション指定の意味で！が表示されます。）
>W/S 10	（上記のエミュレーションの中断を２５０ｍｓｅｃの１０倍のタイミングで行います。初期値は１です。）
>W/S	（現在の倍数の値を表示します。）

書式 (1) __WATCH　［<ウォッチ行>・・・］
(2) __WATCH　＊

＊（アスタリスク）：　すべてのウォッチ行の削除

機能　ウォッチウィンドウに登録したウォッチ行を削除します。

解説　ウォッチ行を削除するには、ウォッチウィンドウの左端に表示されているウォッチ行番号を指定してください。

＜ウォッチ行＞を省略した場合には、最後に登録したウォッチ行を１行削除します。

すべてのウォッチ行を削除する場合には、アスタリスク '＊' を指定してください。

例

ウォッチ行の２行目と４行目を削除する場合。

```
>_WATCH 2 4
```

ウォッチ行をすべて削除する場合。

```
>_WATCH *
```

１０．エラーメッセージ

　本ＩＣＥは、エラーメッセージを日本語で表示しますが、注釈を必要とするエラーメッセージとその対策方法を下記に示します。

・コンフィグレ－ションファイルの内容が不当です。
・コントロールボードが実装されていません。

「2.4 コンフィグレーションの設定」を参照して正しい内容に修正してください。

・指定したアドレスが不当です。

ＬＯＡＤコマンドにおいて表示されたときは、ロードモジュールのロードアドレスが不当です。

・指定したファイルはオープンできません。

コマンドファイル、またはＡＵＴＯ　コマンドで、コマンドファイル実行のコマンド　＜　を実行しようとしたときも表示されます。

・ロードモジュールのフォーマットが不正です。

ＬＯＡＤ、ＶＲＦＹコマンドで指定したオプションと、ロードモジュールのフォーマットが一致しているかどうか確認してください。

・サービスを作成できません－指定されたサービスは既に開始されています。

ＩＣＥ起動時に上記のメッセージボックスが表示された場合は、［ＯＫ］ボタンを選択した後、再起動してください。

ＭＹ－ＩＣＥ　ＡＥ
Ｈ８／３００Ｈ　シリーズ
モニタ　ｆｏｒ　Ｗｉｎｄｏｗｓ９５／ＮＴ
オペレーションマニュアル
第３版
発行年月日　平成１２年１月
発行　株式会社　日立超ＬＳＩンシステムズ

株式会社　日立超ＬＳＩシステムズ

本　　　社　〒187-8522 東京都小平市上水本町 5-22-1　　TEL 042(326)1111

技術サポートセンタ
〒183-0044 東京都府中市日鋼町 1-1 Ｊタワー（9F）　　TEL 0120(25)3047
FAX 042(351)6609

修理サービスセンタ
〒183-0044 東京都府中市日鋼町 1-1 Ｊタワー（9F）　　TEL 042(351)6663
FAX 042(351)6664

東京営業所
〒183-0044 東京都府中市日鋼町 1-1 Ｊタワー（9F）　　TEL 042(351)6600
FAX 042(351)6601

（受付時間 10:00-11:30/13:00-16:30）